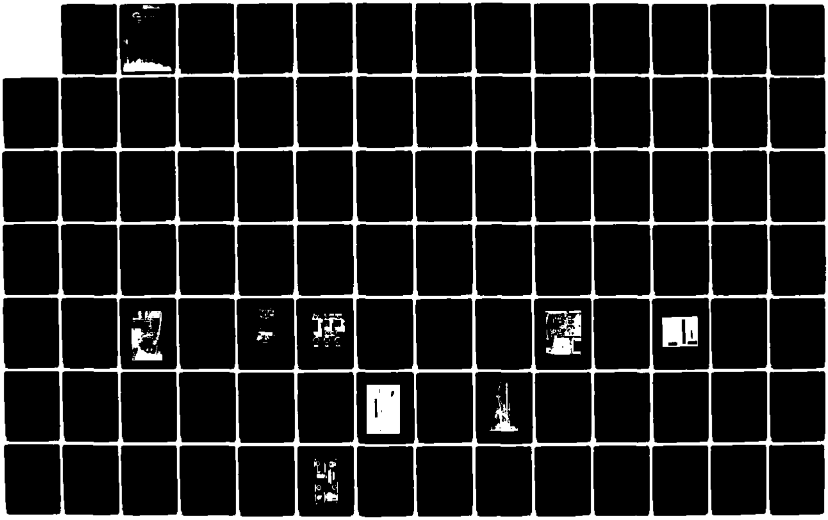
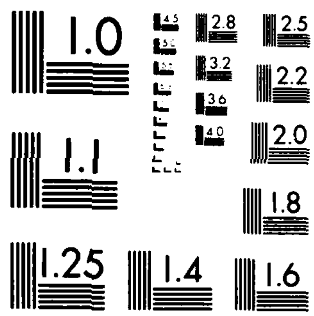
AD-A144 136 DEMAND CONTROLLED ECONOMIZER CYCLES: A DIRECT DIGITAL 1/3
CONTROL SCHEME FOR..(U) AIR FORCE INST OF TECH
WRIGHT-PATTERSON AFB OH S T TOM MAY 84
UNCLASSIFIED AFIT/CI/NR-84-33D F/G 13/1 NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963-A

AD-A144 136

RAY W. HERRICK LABORATORIES

A Graduate Research Facility of The School of Mechanical Engineering

DEMAND CONTROLLED ECONOMIZER CYCLES: A DIRECT DIGITAL CONTROL SCHEME FOR HEATING, VENTILATING, AND AIR CONDITIONING SYSTEMS

Sponsored by

Physical Plant
Purdue University

Internal HL 84-10

Purdue University
West Lafayette, Indiana 47907

1

DEMAND CONTROLLED ECONOMIZER CYCLES: A DIRECT DIGITAL CONTROL SCHEME FOR HEATING, VENTILATING, AND AIR CONDITIONING SYSTEMS

DEMAND CONTROLLED ECONOMIZER CYCLES: A DIRECT DIGITAL CONTROL SCHEME FOR HEATING, VENTILATING, AND AIR CONDITIONING SYSTEMS

Sponsored by

Physical Plant
Purdue University

Internal HL 84-10

Submitted by:

Steven T. Tom, Graduate Research Assistant
Keith Hawks, Principal Investigator

Approved by:

Raymond Cohen, Director
Ray W. Herrick Laboratories

May 1984

UNCLASS

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER AFIT/CI/NR 84-33D	2. GOVT ACCESSION NO. AD-A144136	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Demand Controlled Economizer Cycles: A Direct Digital Control Scheme for Heating, Ventilating, and Air Conditioning Systems		5. TYPE OF REPORT & PERIOD COVERED ~~THESIS~~/DISSERTATION 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Steven T. Tom		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS AFIT STUDENT AT: Purdue University		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS AFIT/NR WPAFB OH 45433		12. REPORT DATE May 1984 13. NUMBER OF PAGES 244
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASS 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

APPROVED FOR PUBLIC RELEASE: IAW AFR 190-17

Lynn Wolaver 5 Jul 84

LYNN E. WOLAVER
Dean for Research and Professional Development
AFIT, Wright-Patterson AFB OH

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

ATTACHED

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASS

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

ACKNOWLEDGEMENTS

It would be impossible for me to list everyone who has given me guidance and assistance in the course of this experiment. At the risk of omitting some names; however, I must offer special thanks to those who have helped me the most. First and foremost of these is Professor Keith H. Hawks. He gave me the encouragement I needed to persevere as well as the technical advice I needed to proceed.

Special thanks are also due to the Purdue Physical Plant. In addition to providing materials to support this project, they gave me invaluable advice and allowed me to experiment with their buildings. Joe Mikesell, Chuck Plantenga, Harold Summers, and the entire Temperature Control Shop were especially helpful.

Larry Stair and his Electronics Shop staff provided me with much of the equipment I needed to conduct this experiment, and Jim Baer of Herrick Labs performed the nearly impossible feat of transforming my vague notions of electronics into functioning hardware.

The Civil Engineering faculty of the Air Force Institute of Technology at Wright-Patterson Air Force Base gave me the technical background I needed for this project and, perhaps even more importantly, their infectious enthusiasm for HVAC controls inspired me to direct my research into this field. Special thanks are also due to the Air Force as a whole for providing me with the opportunity and the funding to pursue my degree.

I wish to thank my family for the moral support they gave me and give special thanks to my fiancee, Elizabeth Keeler. Betsy not only tolerated my incessant "shop talk", she gave me her professional service as a technical librarian, proofread my thesis, and gave me encouragement and faith throughout my research.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Tom, Steven Treece. Ph.D., Purdue University, August 1984. Demand Controlled Economizer Cycles: A Direct Digital Control Scheme For Heating, Ventilating, And Air Conditioning Systems. Major Professor: Keith H. Hawks. Department Of Mechanical Engineering.

Conventional economizers admit excess outside air into a Heating, Ventilating, and Air Conditioning (HVAC) system whenever this air is cool enough to reduce the air conditioning load. When these economizers are used with dual duct, multi-zone, variable air volume, or any other type of HVAC system which uses a common air supply for the heating and cooling coils, the admission of outdoor air has the undesirable side effect of increasing the heating load. The economizer control must therefore balance the increased heating cost against the decreased cooling cost when deciding whether or not to admit this cool air. Conventional economizers base this decision on the outdoor air temperature, but often this is not a reliable indicator of the actual building loads.

In this experiment an improved economizer control was developed which based the control decision on the measured demand for hot and cold air. The control system was

activated in January of 1984 and was operated through February, March, and April. Data taken during these months showed the experimental control system reduced the coil operating costs by over 20% during February and March, and reduced costs by over 30% during April. A computer simulation of this system predicted an annual savings of 22%, or approximately $2200 for the HVAC system being studied. The payback period for the demand controlled economizer ranged from 6 months (if added to an existing DDC system) to 2 years (if installed as a stand-alone system).

In addition to studying the performance of the experimental control system, the instrumentation installed as part of this experiment allowed the entire HVAC system to be studied. Temperature sensors, coil stratification, and perimeter heating systems are among the topics discussed in this thesis.. The calculation of the coil loads required the airflow through the coils be known, so flow sensors were installed in the ductwork. Heated thermistor sensor were used for this purpose, and since these sensors are not commonly found in HVAC systems particular attention was paid to their performance. It was found that they performed quite well, and provided invaluable data as to how the entire HVAC system was operating. The use of these sensors may prove to be valuable even in systems which do not use a demand based economizer.

INTRODUCTION

Commercial Heating, Ventilating, and Air Conditioning (HVAC) systems account for a significant portion of the energy consumed by this country. Estimates of this consumption range between 4% (1) and 16% (2) of the total US energy use, so conservation efforts in this field are vital to our national interests. On a more immediate level, owners of commercial buildings have seen their operating costs skyrocket with the rise in energy costs and money which should have been spent on more productive endeavors has been wasted on inefficient HVAC systems. Many of the systems currently in use were designed when energy was cheap, and efficient use of energy was not a primary goal in their design. The decades of cheap energy also did not encourage adequate maintenance, and the HVAC systems of some buildings have been allowed to deteriorate to the point where the system is using much more energy than even a poor design would dictate. For this reason the first step which any building owner should take to improve his HVAC systems is to inspect the equipment and get it operating the way it was originally designed. The results of such a program can be dramatic, savings of over 50% are not unusual. (3) After these "quick fix" savings have been achieved, additional savings can be obtained through minor modifications and major retrofitting. One energy saving technique which has recently gained popularity in both new construction and as a modification to existing systems is the use of Direct Digital Controls (DDC). A DDC unit is basically a digital computer which controls various elements of a HVAC system directly, as opposed to a supervisory computer system which merely monitors or adjusts a separate system of conventional controls. DDC units can vary in size from a small microprocessor which controls a specific function to a mainframe computer which controls an entire campus. Because they can implement complicated control schemes and provide much more precise control than is possible with conventional pneumatic or electric controls, they can often improve the operating efficiency of existing systems at a fraction of the cost of equipment replacement. The control scheme described in this thesis is an example of a strategy which is particularly effective with existing dual duct (or multizone) systems. It is not intended to justify the cost of a new DDC system by itself, but is offered as a strategy which can increase the efficiency of many HVAC systems which utilize commercially available DDC units.

1.1 Typical Dual Duct System

A schematic of a typical Dual Duct HVAC system is shown in Figure 1.1. Air in the system circulates through what is essentially a closed loop, with a certain percentage of the air being exhausted to the atmosphere and an equal amount of outdoor air being brought in to the loop. This mixture of air returning from the building's rooms or zones (a zone is several rooms with similar loads which are controlled by a single thermostat) and outdoor air is forced through a filter and then separates into two paths. One path, called the hot deck, includes a heating coil and thermostatic control to maintain the air in this path at an elevated temperature, typically around 80 degrees Farenheit (80 F).. The other path is called the cold deck and chills the air to around 60 F. From the hot and cold decks a network of ducting leads the heated and chilled air to each individual zone. Here the zone thermostat controls the supply air temperature through dampers which mix hot and cold air as required to maintain the desired room temperature. The temperature of this supply air varies with the room load. If the room load is negative (i.e. the zone is losing heat to the environment, as on a cold winter day) the supply air temperature must be higher than the desired room temperature to offset this load. If the zone load is positive the supply air must be colder than the room setpoint. The flow of supply air (measured

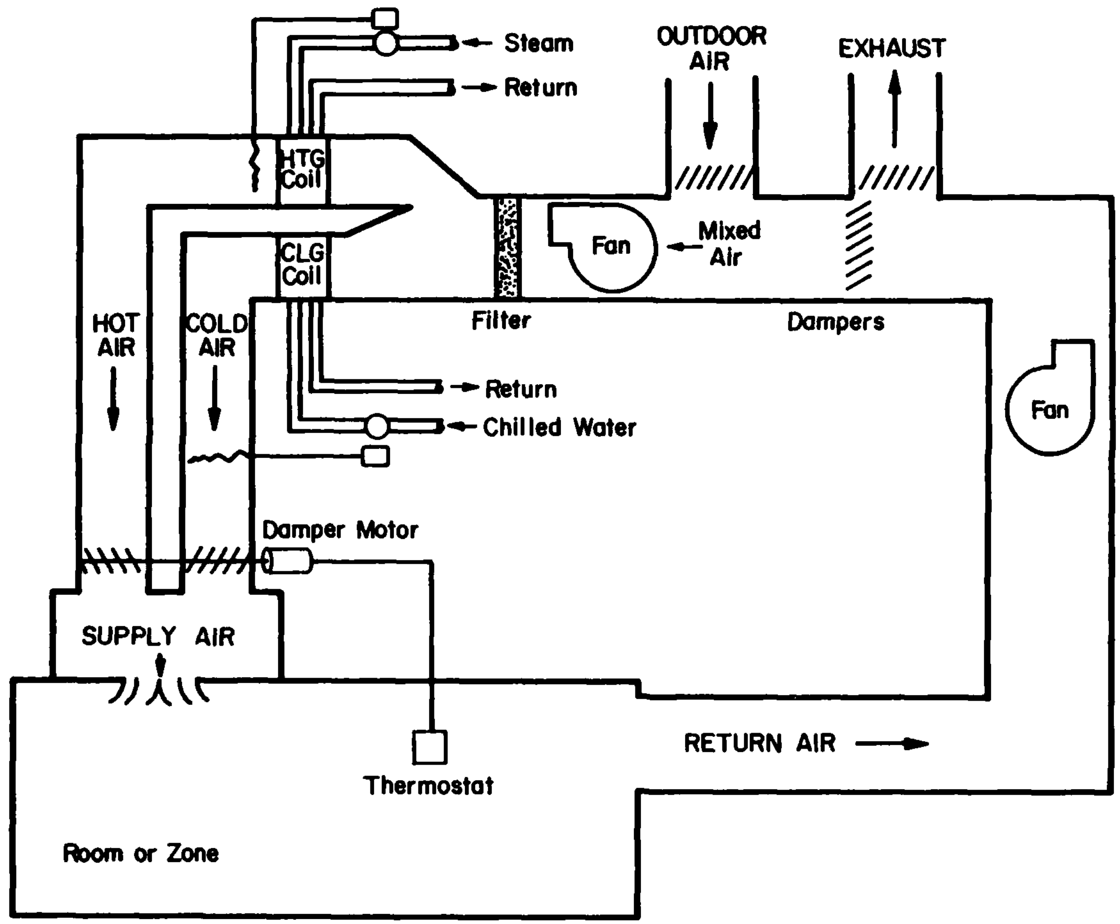


Figure 1.1
Typical Dual Duct System

in cubic feet per minute or cfm) is essentially constant, the temperature varies to meet the load. A flow of air equal to the supply flow is exhausted from the zone and routed through the return air ductwork (typically hallways and stairwells are used as return ducts). The return ducts lead to the mixed air section, and the circuit is repeated.

A dual duct HVAC system usually provides good control of space temperatures because each individual room or zone can mix the hot and cold air to meet its specific load. The hot and cold decks and the ductwork are sized so that even under extreme conditions, with all zones calling for maximum heating or cooling, the system can supply sufficient hot or cold air to meet the demand. As a rough estimate, it has been stated that a typical dual duct system operates at design conditions approximately 5% of the time and requires simultaneous heating and cooling 95% of the time. (4) Since the system requires simultaneous heating and cooling so much of the time, it is not particularly efficient. Irreversible energy losses occur when the hot and cold airstreams are mixed, and the physical layout of the hot and cold decks often allows undersirable heat transfer between the two. Because of their good control and moderate first costs they were very popular in buildings designed prior to the current energy consciousness and will be in service for many years to come.

Purdue, for example, has 52 dual duct systems in service in various campus buildings. The research described in this thesis was done on a dual duct system; however, the mixed air section of multizone systems and some variable air volume (VAV) systems are essentially identical to that of a dual duct system and the results obtained should be equally valid for those systems.

1.2 Conventional Mixed Air Section

The mixed air section is where the return air from the individual rooms or zones is mixed with outdoor air before entering the hot and cold decks. A minimum flow of outdoor air must be admitted to maintain acceptable air quality in the zones served by the system. In non-smoking office areas, for example, the ASHRAE (American Society of Heating, Ventilating, and Air Conditioning Engineers) standard is 5cfm of outdoor air per occupant. (5) Heating and cooling this outdoor air can often cost $.50 to $.75 per cfm per year (6) so the cost of bringing outdoor air into a large commercial building can be enormous. For this reason the Energy Research and Development Administration (ERDA, now the Department of Energy) listed control of ventilation air as its number 2 priority for energy conservation, behind the survey of control related losses. (2) As an example, the Krannert building, where this research was performed, is served by six separate

HVAC systems which require a total of 37,850 cfm outdoor air to meet minimum ventilation requirements. The mechanical systems in this building will be described more completely in chapter 2, the point to be made here is that it costs $26,500 per year just to treat this outdoor air as it is brought into the system. (This cost is based upon estimates made by the Purdue Physical Plant, which is the department of the university which designs, builds, and maintains all buildings on Purdue.) During moderate weather, however, it is sometimes possible to use this outdoor air to help cool the building, so more than just the minimum amount should be brought in to the system.

A conventional mixed air control scheme is shown in Figure 1.2. In this system a mixed air controller feeds a signal to the damper motors to maintain a fixed mixed air temperature, 60 F in this case. A rise in mixed air temperature will cause an increase in the controller's output (assuming a direct acting controller) which will cause the outside and exhaust air dampers to open and the return air dampers to close. This increases the percentage of outdoor air in the mixed air. If the outdoor air is cool enough, the mixed air controller will be able to maintain the 60 F setpoint. If the outdoor air is above 60 F, the controller will not be able to maintain setpoint and will force the dampers to a position which admits 100% outdoor air. When the outdoor temperature rises above a

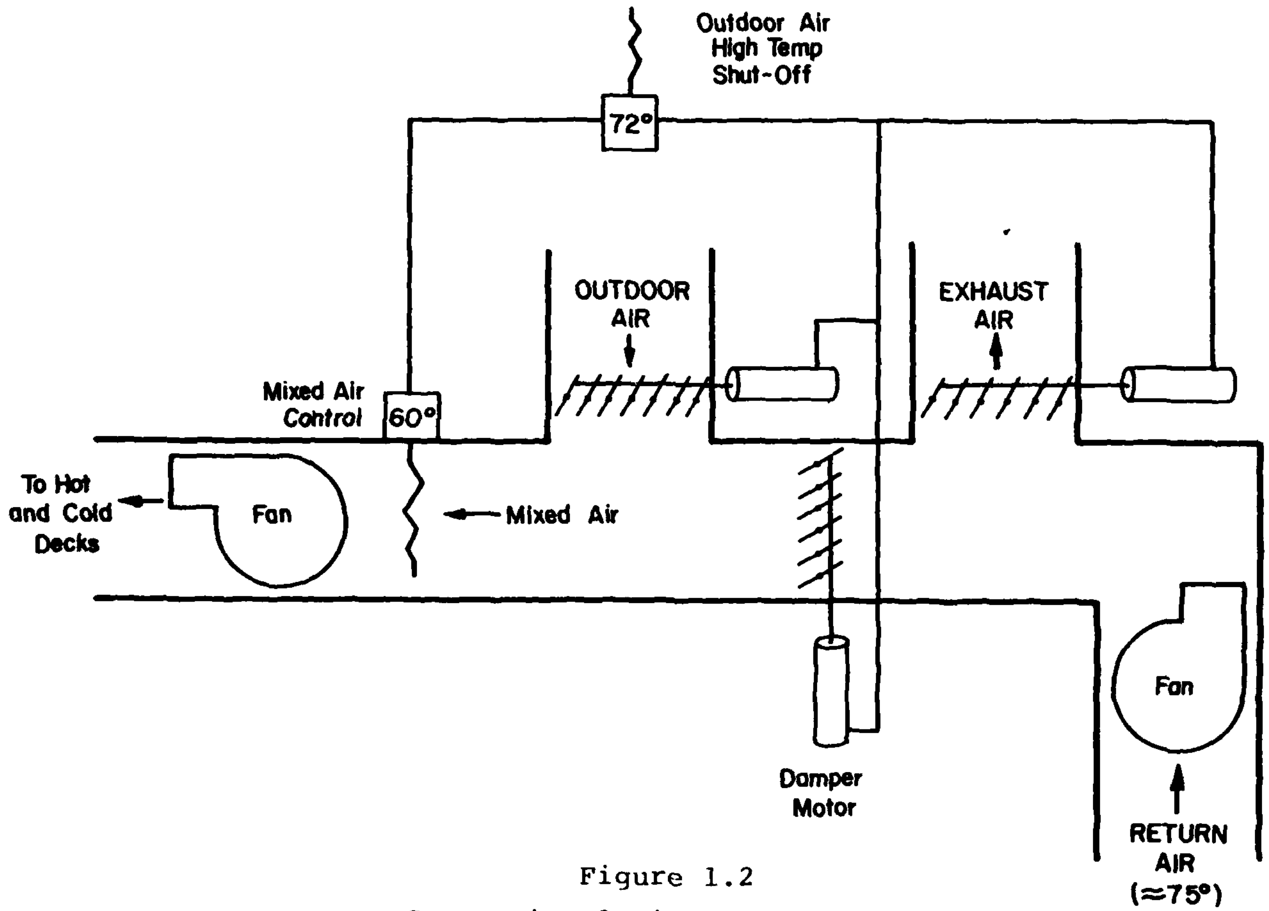


Figure 1.2
Conventional Mixed Air Control

fixed cut-off temperature, 72 F in this case, a switch shuts off the signal to the dampers and causes them to close to the minimum position, since the outdoor air is too hot to provide any cooling. (The required minimum air intake is provided by mechanical stops on the dampers, by separate dampers, or by other means.) Sometimes a similar switch is provided to shut the dampers if the outdoor air gets too cold, since it is assumed that there is then no strong demand for cooling, but since only a small percentage of very cold outdoor air is required to maintain a 60 F mixed air setpoint this switch is often omitted. This type of mixed air control is commonly known as an Economizer, or as a Dry Bulb Economizer since the control is based upon dry bulb temperatures. A similar type of control which measures both the temperature and the humidity of the various airstreams and controls based upon the total heat contents is commonly known as an Enthalpy Economizer, or simply as Enthalpy Control.

The main problem with either dry bulb or enthalpy economizers is that they base their control solely upon the outside weather conditions and cannot adjust for other factors which affect the building load. The amount of cooling a well insulated building requires is usually influenced more by people and heat generating equipment inside the building than it is by outside temperatures, and buildings with large glass exposures can be similarly influenced by solar gains. All of these factors can change drastically from day to day, so control systems based solely upon outside air conditions can make many mistakes. During a mild winter day, for example, a building with only a small internal load would primarily require heat and opening the outdoor air dampers past their minimum would be very expensive. If a large number of people suddenly entered the building (as at 8:00 am on a weekday, for example) the building would then have a large cooling load and not opening the dampers would be expensive. Conventional economizer controls cannot adapt to this changing load and often fail to provide the savings expected. A leading HVAC controls manufacturer states that an enthalpy economizer is usually the wrong answer for a dual duct system (7), and an internal Purdue Physical Plant report only recommends economizers if the system cooling load is at least 75% of maximum capacity during April through October. Many dual duct systems do not have economizers installed, and many systems which do have economizers have had them disconnected.

1.3 Experimental Mixed Air Control

Since the amount of outside air required for efficient operation is determined by the heating and cooling loads on the building, a microprocessor can be used to measure this demand and control the mixed air dampers accordingly. A schematic of this control system is shown in Figure 1.3. Temperature sensors on either side of the heating and cooling coils give the temperature difference across each coil. Since the flow through each coil varies with the heating and cooling loads on the building, flow sensors are installed in the hot and cold decks. The microprocessor combines the temperature and flow data to calculate the energy used by each coil. Since the goal of energy conservation in HVAC systems is primarily to reduce operating costs, the microprocessor multiplies the energy figures by their respective costs to calculate how much money is being spent to heat and cool the mixed air. By measuring the temperatures of the outdoor air and the return air, the microprocessor can calculate what the mixed air temperature would be with varying percentages of outdoor air, calculate what the heating and cooling costs would be for each percentage, and select the mixed air temperature which gives the lowest total operating cost. The microprocessor then generates a control signal which will adjust the dampers to provide this optimum mixed air temperature. The use of heating and cooling costs ($/hr) rather than energies (btu/hr) can have a significant effect on the savings available. At Purdue, for example, it costs twice as much to chill 1 cfm of mixed air 1 degree F as it does to heat it 1 degree F, so a controller

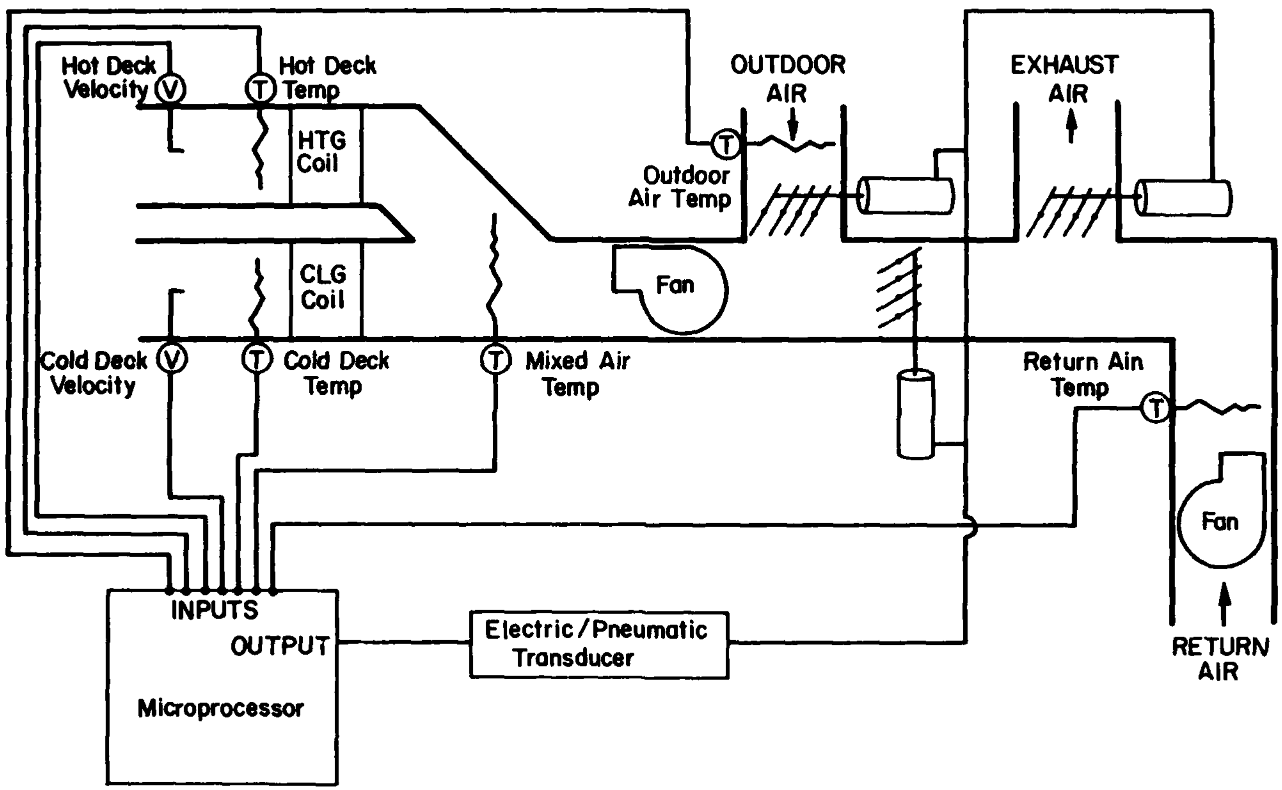


Figure 1.3
Demand Based Economizer

which minimizes cooling energy will operate more economically than one which minimizes total energy.

1.4 Related Work By Others

The control strategy just described is unique in that it measures the actual heating and cooling loads on the building and uses these loads to calculate the optimum mixed air temperature. It is also unique in that it uses operating costs rather than energy to determine this optimum. Nestor (8) suggested adjusting the outdoor air cut-off temperature of a conventional economizer based upon the hot and cold deck loads, but I can find no evidence that his idea was ever acted upon. McKew (9) suggests using separate mixing chambers for the hot and cold decks so that the cold deck could operate in an economizer mode year round while the hot deck never operates on more than the minimum amount of outdoor air. This idea is inherently superior to the microprocessor control scheme but is not readily adaptable to existing HVAC systems. Lambert and Engineer (10) propose a control scheme which varies the minimum percent of outside air admitted throughout the day in accordance with new ASHRAE guidelines (11), but this scheme is intended to minimize energy usage when economizer operation is not feasible. Their scheme could be mated with the one described in this thesis to provide year round savings. Janeke (12) suggests several novel uses for free cooling and better methods of exhausting room air but since his ideas are not readily adaptable to existing structures they would be better combined with those of McKew than with this controller. Kallen (13) gives some of the pros and cons of existing economizer systems during the "shoulder season", i.e. times of the year when the system is operating at neither maximum heating or cooling. Several researchers are working on measuring the contaminants present in the return air and determining the minimum percent outside air required to dilute these contaminants (again in accordance with new ASHRAE guidelines), and this type of measurement should be fully compatible with the scheme described in this thesis.

1.5 Objective

The primary purpose of this research was to demonstrate the feasibility of the control scheme by taking measurements of its use in an operating HVAC system. Many Direct Digital Control (DDC) units already on the market could probably be modified to implement this control scheme, so the development of commercially marketable equipment was not a goal of this research. The equipment used in this project was selected for its availability and expediency, rather than for its suitability for operational (i.e. non-laboratory) use.

PRELIMINARY STUDIES

2.1 Test Site

The site chosen as a test site for the load based economizer was the Krannert building on the main campus of Purdue University, West Lafayette Indiana. This seven story building, completed in 1964, houses Purdue's school of management. The system used in this test was unit number ACP-7, which supplies rooms on floors 2 through 7 in the central section of the east wing of the building. A copy of the floor plans for the Krannert building is given in the appendix, shaded rooms indicate those served by ACP-7. Most of the rooms served by this system are offices for staff and graduate assistants. On the second and third floor a portion of the Krannert Library is served by ACP-7, and on the 7th floor a small computer room (desktop computers) and a portion of the Pharmacy lab are on this system.

System ACP-7 itself is a typical dual duct system as described in the previous chapter. The supply fan is rated at 17,355 cfm, and the minimum outside air required is 3,000 cfm or 17% of the supply cfm. The cold deck houses a 48 ton (576,000 Btu/hr) cooling coil and the heating coil will handle 612,000 Btu/hr. A schematic of the controls for this unit (including the interface with the microprocessor damper controls) is given in the appendix as Figure A.5. The original design for the building called for a conventional economizer capable of bringing in 100% outside air, however the dampers which were eventually installed allow for only 85% outside air. Originally a small damper controlled by an electric to pneumatic relay opened fully to admit minimum outside air whenever the fan was turned on. A pneumatic controller activated large economizer dampers to maintain a mixed air setpoint of 60 F (with a 74 F outside air shut-off); however, this economizer was disconnected in 1976 (approximately). Calculations made at that time using contemporary energy cost figures had shown potential savings to be minimal. Several cooling coils at Purdue had frozen in past winters because of dampers which admitted too much outside air, so many economizers were disconnected at about that time. The heating and cooling coils are controlled by pneumatic controllers. The cold deck controller maintains the cold deck at a constant setpoint, and the hot deck setpoint is reset based upon outside air temperature. If it operated as designed, this controller would maintain the hot deck at 80 F when the outside air temperature is -10 F or below and would gradually reduce the hot deck temperature to a minimum of 70 F when the outside air warms to 60 F. In practice, the hot deck temperature is maintained at higher temperatures to combat stratification. (The hot deck splits into two supply ducts immediately downstream of the coil, and the duct closest to the coil inlet carries air that is as much as 15 degrees warmer than the duct at the other end of the coil. Thus to keep the cooler duct at 80 F the other duct has to be maintained at 95 F and the average hot deck temperature is therefore about 88 F.) Also, the reset schedule is somewhat erratic and unpredictable. The effects of these problems will be discussed in more detail later. The poor performance of the reset controller is the exception rather than the rule, as Physical Plant personnel have maintained system ACP-7 in excellent condition.

2.2 Computer Simulations

2.2.1 Dry Bulb Economizer: Program Pdeck

A computer program titled "Pdeck" was written to test the basic decision making algorithm and to predict the performance of an ideal economizer under local weather conditions. A copy of this program is given in the appendix and flow charts showing the basic structure are given in Figures 2.2 through 2.4. The program itself is very straightforward and only a few key calculations will be described here. The program uses a very simple model to simulate the performance of ACP-7, and uses "bin data" to predict outside weather conditions throughout a 12 month period. The bin method is a procedure for averaging weather data collected over an extended period, typically 10 or 20 years. The data are sorted into 5 degree "bins" for each month based on outside air temperature, and the average wet bulb temperature together with the hours duration are recorded for each bin. As an example, during the month of May the outdoor temperature ranges between 55 and 59 F for an average of 105 hours, and the average wet bulb temperature for this bin is 51 F. The bin data used was collected by the U.S. Air Force at Grissom AFB In. (14), a site approximately 30 miles from Purdue. In program Pdeck the midpoint of each bin is used, i.e. the program calculates the performance of system ACP-7 given outdoor conditions of 57 F dry bulb and 51 F wet bulb and assumes these conditions occur for 105 hours in May.

The original design calculations for the Krannert Building are no longer available, so a rough estimate of the heating and cooling loads on system ACP-7 was made from the coil capacities. Using design conditions commonly used in the sixties, it was estimated that the coils were designed to satisfy a sensible heating load of 273,400 Btu/hr, a latent load of 45,600 Btu/hr, and a U-factor through the building walls of 5,080 Btu/hr F.

Thus for any known outside and room air temperatures the sensible and latent loads on system ACP-7 could be calculated as:

$$Q_S = 273{,}400 \text{ B/hr} + (5{,}080 \text{ B/hr F})(T_{OA} - T_{RA}) \quad (2.1)$$

$$Q_L = 45{,}600 \text{ Btu/hr} \quad \text{(fixed)} \quad (2.2)$$

where:

Q_S = Room Sensible Load (Btu/hr)

Q_L = Room Latent Load (Btu/hr)

T_{OA} = Outside Air Temperature (F)

T_{RA} = Room Air Temperature (F)

Note that the equations given above describe a load which does not vary throughout the day and which depends only upon outside weather conditions. Also, the bin data used simulates 24 hour a day operations whereas system ACP-7 is operated on a schedule which varies with the day of the week and is altered for holidays and unusual weather. Program Pdeck was therefore not expected to provide accurate predictions of the actual cost of operation, but instead was used to develop a decision making algorithm and to give a "ballpark estimate" of efficiency. The program relies on a subroutine called "psyc" (written by others prior to this research) to calculate the psychrometric data of the air at various points in the air conditioning cycle.

A psychrometric chart showing the heating, cooling, and mixing processes occurring within system ACP-7 is given in Figure 2.1. This chart shows conditions which might occur on a day when economizer operation was feasible. The actual temperatures and humidities shown are not important, in fact the scaling has deliberately been distorted to "expand" the drawing for the purpose of illustration. The chart is merely intended to help explain the calculations which are being made by program Pdeck. The mixing and process lines on this chart will be described with the sections of program Pdeck which implement them.

Basic Structure:

As shown in Figure 2.2, program Pdeck begins by setting various parameters to their initial values and by reading bin data for one bin. The outside air conditions (enthalpy, specific humidity, etc.) are calculated for this bin using subroutine psyc and the percent of outside air admitted to the system is set at its minimum. An initial value for the return air temperature and humidity is assumed from the design conditions and the resulting flows and conditions throughout the system are calculated for this assumption. (Note that throughout these calculations it is assumed that the room air and return air are identical, i.e. no heating or cooling takes place in the return duct. Also the entire system of rooms supplied by ACP-7

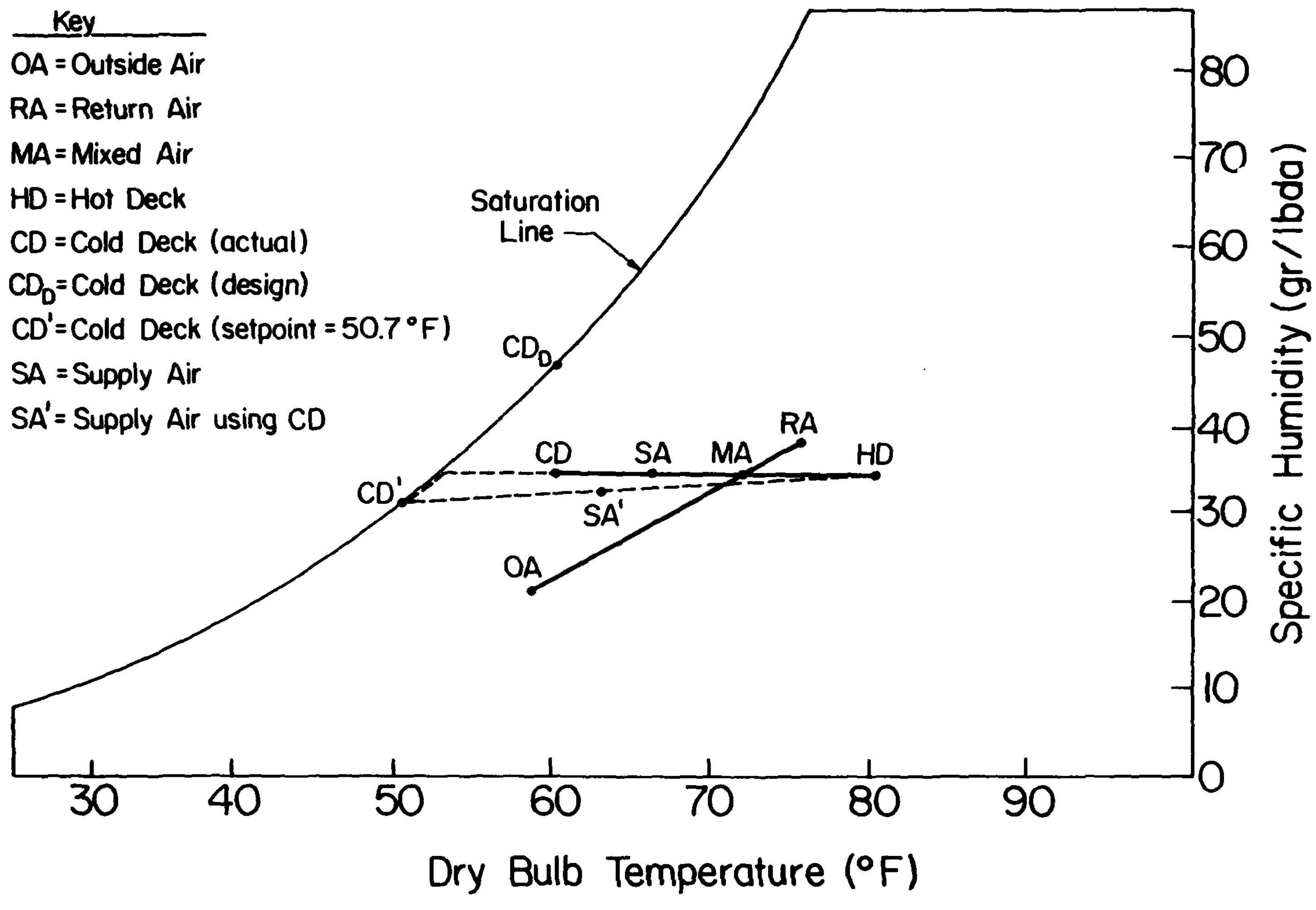


Figure 2.1 Psychrometric Chart

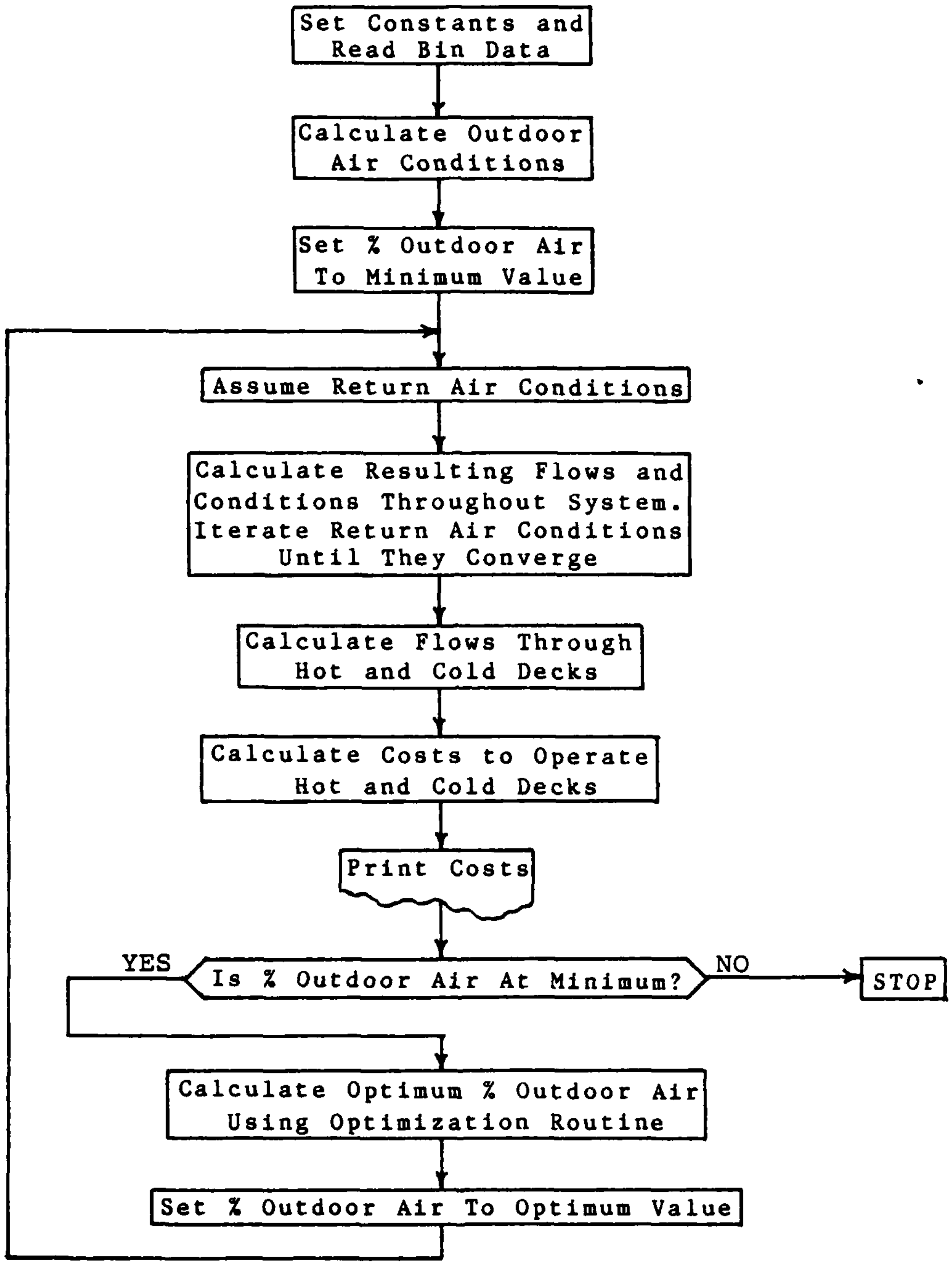


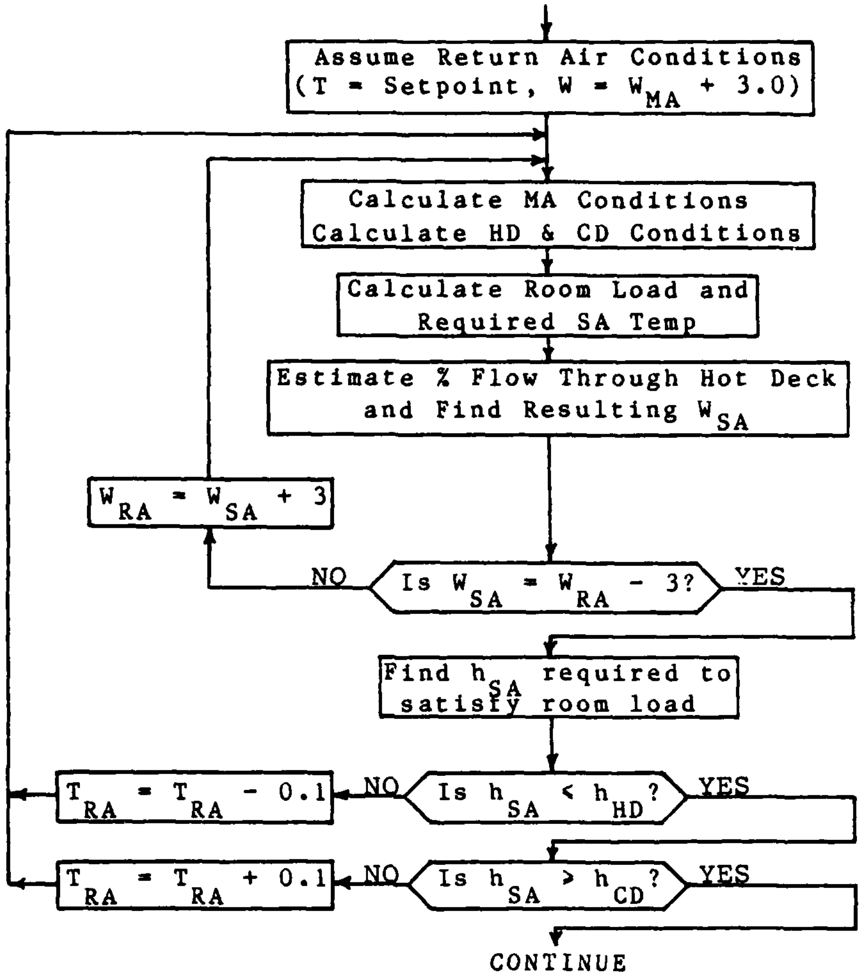
Figure 2.2

Basic Flow Chart For Program PDECK

is treated as a single room with a load equal to that of the system.) These steps eventually lead to a calculated return air condition which is compared to the initial estimate. A revised estimate is then prepared and the iteration is repeated until it converges. This iteration process is shown in Figure 2.3 and will be described later. Once the basic conditions throughout the system have been fairly well established by the iteration process, the flow of hot and cold air needed to meet the room load is found by comparing the required supply air enthalpy to the hot and cold deck enthalpies. Given the flows and enthalpy changes across the coils, the energy used by each coil can be calculated and converted to costs ($/hr) using price figures supplied by Purdue Physical Plant. Having established the cost of running the system on minimum outside air, program Pdeck performs an optimization algorithm and repeats the cost calculations using optimum outside air.

<u>Iteration loop</u>:

The heart of program Pdeck is the iteration loop which calculates the conditions throughout the air conditioning cycle. This loop is shown in Figure 2.3. The outdoor air conditions are known from the bin data, and are shown as point "OA" on Figure 2.1. The first step is to make an assumption as to the return air conditions. If



Superscripts

T = Temperature (Deg. F)
W = Humidity (gr/lbda)
h = Enthalpy (Btu/lbda)

Subscripts

RA = Return Air
MA = Mixed Air
SA = Supply Air
HD = Hot Deck
CD = Cold Deck

Figure 2.3

Flow Chart For Return Air Iteration

the room load does not exceed the coil capacities the room air will be at the room setpoint (75 F). The latent load given in eqn. 2.2 translates into a humidity gain of 3 grains per pound of dry air (gr/lbda, 7000 gr = 1 lb) at the room design conditions, so program Pdeck assumes for a first guess that the return air humidity is equal to the supply air humidity plus 3 gr/lb. It also assumes that no condensation takes place in the cooling coil, so the specific humidity of the supply air entering the room is equal to the specific humidity of the mixed air entering the hot and cold decks. Therefore the first iteration assumes:

$$W_{RA} = W_{MA} + 3\text{gr/lb.da} \qquad 2.3$$

where

W_{RA} = Specific Humidity of the Return Air (gr/lbda)

W_{MA} = Specific Humidity of the Mixed Air (gr/lbda)

The specific humidity of the mixed air may be calculated using a straight mixing equation:

$$W_{MA} = (W_{OA})(\%OA) + (W_{RA})(100\% - \%OA) \qquad 2.4$$

where of course W_{OA} is the specific humidity of the outdoor air. Combining equations 2.3 and 2.4 and solving for the mixed air conditions yields:

$$W_{MA} = [(W_{OA})(\%OA) + 3.0(100\% - \%OA)] \ / \ \%OA \qquad 2.5$$

The mixed air humidity is now defined solely in terms of the outdoor air humidity and the percent of outdoor air being brought into the system, both of which are known conditions. The humidity of the return air was earlier assumed to be equal to that of the mixed air plus 3 gr/lbda, so it can now be calculated. Both the temperature and the specific humidity of the return air are now known, so the conditions are fully defined. (Point "RA" on Figure 2.1.) The temperature of the mixed air is obtained by a straight mixing of the outdoor and return air temperatures and hence the mixed air conditions are fully defined. (This mixing line is shown between points "OA" and "RA" on Figure 2.1. The figure shows point "MA" lying on this line, 20% of the way from point "RA" to point "OA", indicating the system is operating at 20% outside air.) Once the mixed air conditions are known, the mass flow rate through the system is fixed by the specific volume of the mixed air and the the known capacity (cfm) of the supply fan. This mass flow rate (lbda/min) is considered to be constant, the volumetric flow rates (cfm) at various points in the system are calculated from this mass flow rate once the appropriate specific volumes are found.

Since air passing through the hot deck undergoes sensible heating only, the conditions of the air leaving the hot deck are fully defined as:

$$T_{HD} = \text{Hot Deck Setpoint}$$

and

$$W_{HD} = W_{MA}$$

This is shown as point "HD" on Figure 2.1.

The conditions of the air leaving the cold deck require two calculations since condensation may or may not occur within the deck. Program Pdeck first assumes that condensation does occur and that the air leaves the cold deck at design conditions, namely at 60.7 F dry bulb and 59.4 F wet bulb (unless of course the mixed air entering the coil is colder that 60.7 F, in which case the air leaves the coil at the mixed air temperature). This point is shown as CD_D on Figure 2.1. The psychrometric conditions are calculated for these wet and dry bulb temperatures and the resulting specific humidity is compared to that of the mixed air. If it is lower than the mixed air, then condensation did occur and the assumption is correct. If not, then the cold deck conditions are re-calculated using the specific humidity of the mixed air. For the example shown in Figure 2.1, condensation did not occur and the cold deck conditions are shown as point "CD". If the cold deck setpoint had been, say 50.7 F instead of 60.7 F then condensation would have occurred and the cold deck conditions would have been at the point shown as CD′.

Equation 2.1 is used to calculate the sensible heating (or cooling) load in the room and an estimate of the required supply air temperature is made. This estimate is based upon the assumptions of a constant flow rate through the supply fan and a dry air enthalpy change of 0.24 Btu/lb F Thus:

$$\text{Room Load} = (\text{Airflow})(0.24 \text{ Btu/lb F})(T_{RA} - T_{SA})$$

or

$$T_{SA} = T_{RA} - (\text{Room Load})(V_{MA})/(\text{CFM})(0.24) \qquad 2.6$$

where

Room Load = Load from Eqn 2.1 (in Btu/min)

Airflow = Flow of air through the system (in lbda/min)

T_{RA} = Return (or Room) Air Temp (F)

T_{SA} = Supply Air Temp (F)

V_{MA} = Specific Volume of the Mixed Air (ft^3/lbda)

CFM = Fan flow rate (in mixed air section) (ft^3/min)

Once the required supply air temperature is known, the ratio of hot deck and cold deck flows which will mix to this temperature can be calculated as:

$$\% \text{ HD} = (T_{SA} - T_{CD})/(T_{HD} - T_{CD})$$

where

% HD = Percent of supply air flow which passes through the hot deck, and

T_{CD} = Cold Deck Temperature (F)

Given this mixing ratio, the specific humidity of the supply air can be found as a straight mixing of hot deck and cold deck humidities. On Figure 2.1 this is indicated by the fact that point "SA" must lie on the line between points "CD" and "HD" and must be at the required supply air temperature. Since the latent load in the room raises the supply air humidity by 3 grains per pound as it passes through the room and into the return duct, program Pdeck now checks the initial assumption on the return air conditions. For the example shown in Figure 2.1 the assumption is correct, since no condensation occurred and the supply air is 3 gr/lbda dryer than the return air. If condensation had occurred (say, for example, that the cold deck conditions were at point CD′) then the supply air would be at the conditions shown as SA′ and the return air would no longer be 3 gr/lbda wetter than the supply air. If the difference between the assumed return air humidity and that calculated by adding 3 gr/lbda to the supply air humidity differ significantly (say, by more than 0.5 gr/lbda), a new assumption of:

$$W_{RA} = W_{SA} + 3 \text{ gr/lbda}$$

is made and the cycle is repeated.

Once the humidity calculations converge a more accurate calculation of the required supply air condition can be made by a variation of eqn. 2.6:

$$h_{SA} = h_{RA} - (\text{Room Load})(V_{MA})/\text{CFM}$$

where

h_{SA} = Enthalpy of the supply air (Btu/lbda)

h_{RA} = Enthalpy of the return air (Btu/lbda)

In this calculation the approximation of 0.24 Btu/lb F is avoided since the actual enthalpy "h" (Btu/lb) of the room air is now known. Once the required supply air enthalpy is known it is compared to the available hot deck and cold deck enthalpies to see if the assumption that the room is being maintained at setpoint is justified. If the coils cannot maintain the room at setpoint then a new assumption for the room temperature is made and the iteration is repeated. Otherwise program Pdeck continues with the cost calculations as described in the previous section.

Optimization Algorithm: A flow chart for the routine used to pick the optimum percentage of outdoor air is shown in Fig. 2.4. This algorithm requires the dry bulb temperatures of the outdoor air, return air, hot deck, and cold deck be known, as well as the flow rates through the two decks. The calculations were designed to rely on easily measured data and to be suitable for use in a microprocessor. The first step in the optimization routine is to compare the outdoor air temperature to 72 F and to exit the routine if this temperature is exceeded. The 72 F cut-off is based on the bin data for this area. When the

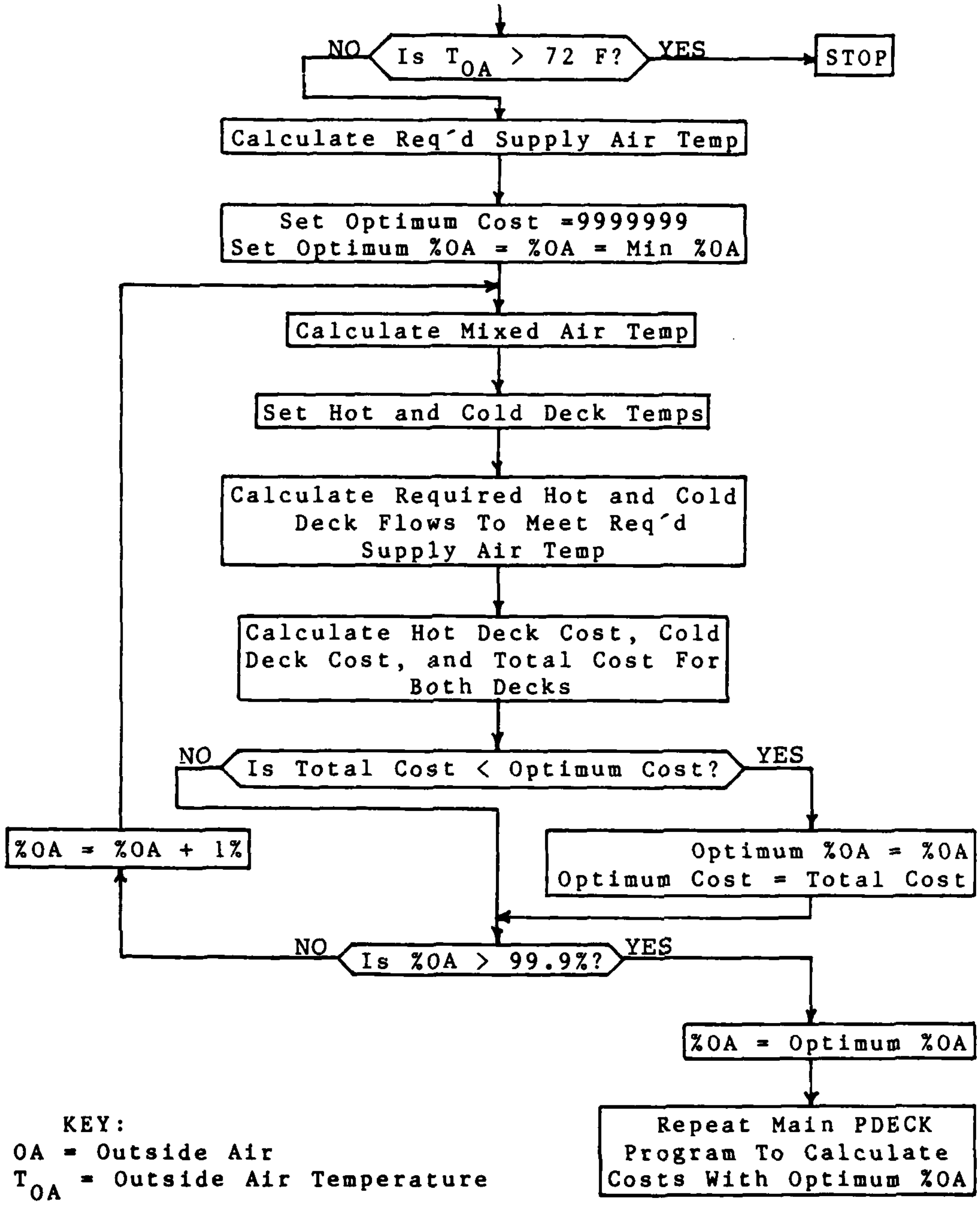


Figure 2.4

Flow Chart For Optimization Algorithm

outside air is warmer than 72 F its enthalpy is generally higher than that of the return air and hence the outdoor air has no cooling value.

If the outdoor air is cool enough to reduce the air conditioning load, the algorithm computes the required supply air temperature from the deck temperatures and flows. (The temperatures and flows are known in this simulation from previous calculations. In the final control application they will be measured.) This is another straight mixing equation:

$$T_{required} = [(CFM_{HD})(T_{HD}) + (CFM_{CD})(T_{CD})]/CFM_{total} \quad 2.7$$

where:

$T_{required}$ = Required Supply Air Temp (to meet room load)

CFM_{HD} = Airflow through Hot Deck (ft^3/min)

CFM_{CD} = Airflow through Cold Deck (ft^3/min)

$CFM_{total} = CFM_{HD} + CFM_{CD}$

T_{HD} = Hot Deck Temp (F)

T_{CD} = Cold Deck Temp (F)

Once the required supply air temperature has been computed various loop parameters are set to their initial values. These parameters include an optimum cost figure, which is initialized to an unreasonably high figure, an optimum percent outdoor air, which is initialized to the minimum percent outdoor air allowable, and a current percent outdoor air variable, which is also initialized to the minimum. The loop then calculates what the mixed air temperature would be using the current percent outdoor air and checks to see if this mixed air temperature would alter the deck temperatures. Once the new deck temperatures are known the routine predicts what the flow through the hot and cold decks will be. Based upon the assumption that the total flow will remain constant:

$$CFM_{CD} = CFM_{total} - CFM_{HD}$$

so Eq. 2.7 can be rewritten as:

$$CFM_{HD} = (CFM_{total})(T_{required} - T_{CD})/(T_{HD} - T_{CD}) \qquad 2.8$$

If changing the mixed air temperature does not affect the deck temperatures the flow rates will obviously be unchanged. If the deck temperatures change, equation 2.8 will predict the new flow rate.

Once the flow rates have been established, the cost (per minute) of running each deck can be estimated. Purdue physical plant records show their steam costs to be $3.90/MBTU and their chilled water costs to be $.079/Ton Hr or $6.58/MBTU. Using design conditions to determine the density of the air leaving the coils and the dry air energy estimate of 0.24 Btu/lb F these costs can be expressed as:

Cold Deck Price:

$$(\$6.5\times10^{-6}/\text{Btu})(0.24\ \text{Btu/lb F})(1\ \text{lb}/13.3\ \text{ft}^3) =$$
$$\$1.19\times10^{-7}/\text{ft}^3\ \text{F}$$

Hot Deck Price:

$$(\$3.90\times10^{-6}/\text{Btu})(0.24\text{Btu/lb F})(1\ \text{lb}/13.7\text{ft}^3) =$$
$$\$5.08\times10^{-8}/\text{ft}^3\ \text{F}$$

Thus the cost of running the system at the current percent outside air is:

$$COST_{HD} = (FLOW_{HD})(T_{HD} - T_{MA})(0.508)$$
$$COST_{CD} = (FLOW_{CD})(T_{MA} - T_{CD})(1.19)$$
$$COST_{TOTAL} = COST_{HD} + COST_{CD}$$

The power of 10 has been dropped from the price figures because they are only being used for comparative purposes. Once the cost of running the system on the current percent outside air is known it is compared to the optimum cost figure. If the cost is less than the optimum, then the current percent outside air stored as the optimum percent outside air. The optimum cost figure is similarly updated, the current percent outside air is increased by 1%, and the loop is repeated until it reaches 100%. At this point the value contained in the optimum percent outside air variable should be the optimum operating point for system ACP-7.

2.2.2 Ideal Economizer: Program Edeck The optimization routine used in program Pdeck (which is fundamentally the same as the one used in the microprocessor control scheme) relies upon several approximations and uses dry bulb temperatures only. To test the validity of these approximations a program called "Edeck" was written which found the optimum operating cost based upon all available information. This program was not written around easily measured variables and was not particularly well suited to microprocessor applications. It was written to model an ideal economizer and thus provide a basis of comparison for the optimization algorithm in Pdeck. A flow chart of this program is shown as Figure 2.5, and a copy of this program is included in the appendix. Basically this program is only a slight modification of Pdeck. Instead of using an optimization algorithm to estimate the best outside/return air mixture, program Edeck performs the system analysis procedure used in the main body of Pdeck to calculate the operating cost for every outside air percentage possible from the minimum allowable to 100%. (in 1% increments) Each time a new optimum cost is found the optimum percent outside air variable is updated. A flag variable is used to control the printed output, the cost figures are printed for minimum outside air and for the optimum percent outside air.

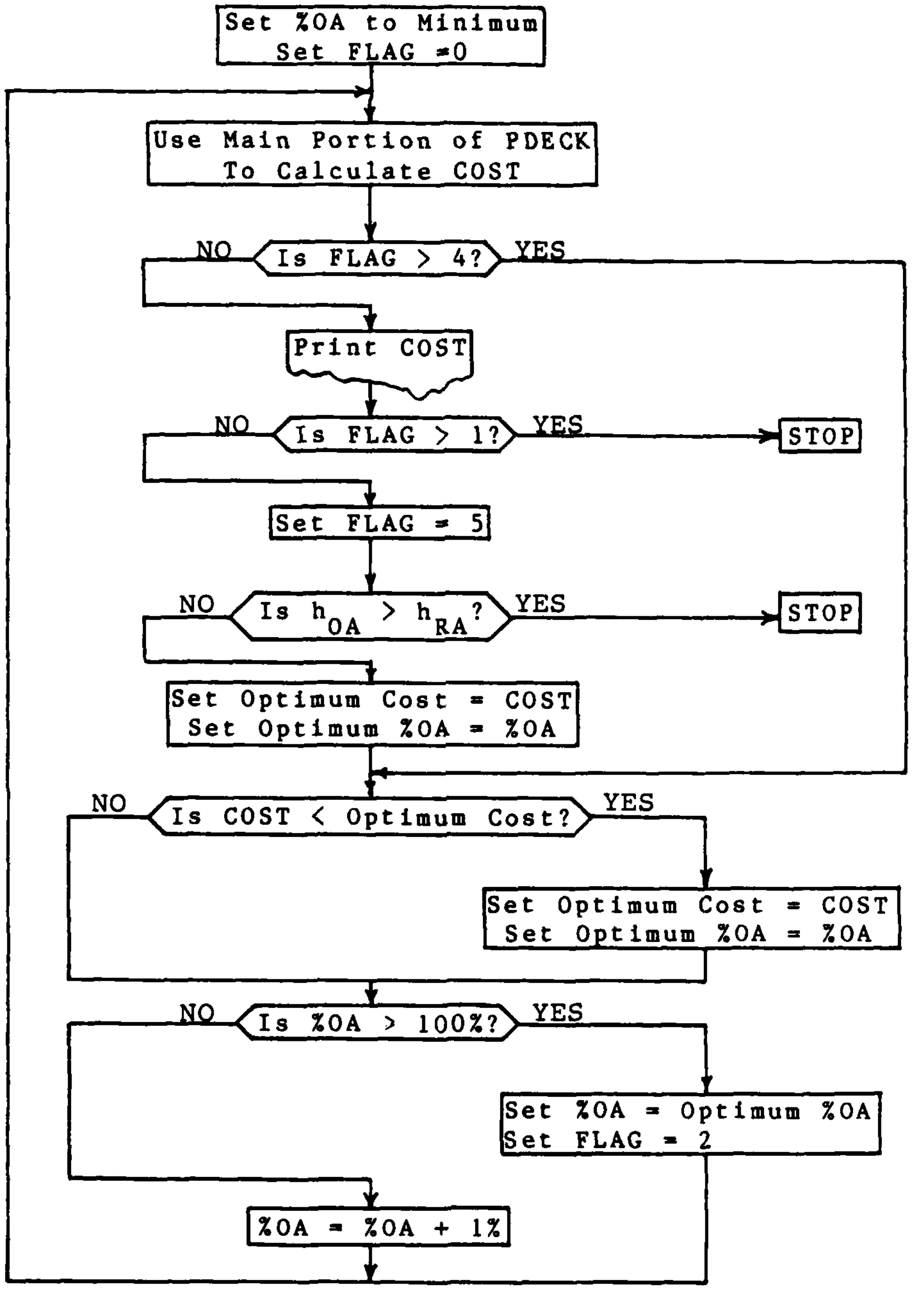


Figure 2.5

Flow Chart For Program EDECK

2.3 Results

The results of programs Pdeck and Edeck are given in Tables 2.1 and 2.3. Since the hot deck reset controller operated erratically, the computer simulations were run once for an ideal hot deck reset and once for a fixed hot deck temperature of 88 F.

Table 2.1

Deck Costs With Hot Deck Reset

(Setpoint varies from 70 F to 80 F)

Costs in dollars for 24 hr/day, 7 day/week Operation									
	On Minimum OA			With Economizer Dry Bulb			Ideal		
	HD	CD	Tot	HD	CD	Tot	HD	CD	Tot
Jan	415	18	433	416	15	431	416	15	431
Feb	324	20	344	326	17	343	325	17	342
Mar	222	91	313	236	56	292	235	57	292
Apr	73	309	382	129	160	289	126	163	289
May	19	648	667	95	371	466	91	372	463
Jun	1	1082	1083	57	806	863	55	808	863
Jul	0	1324	1324	39	1199	1238	38	1152	1190
Aug	1	1236	1237	52	1074	1126	50	1033	1083
Sep	8	854	862	78	600	678	75	603	678
Oct	51	400	451	119	204	323	115	207	322
Nov	164	123	287	187	71	258	185	73	258
Dec	321	38	359	326	28	354	326	28	354
TOT	1599	6143	7742	2060	4601	6661	2037	4528	6565

Total Savings With Dry Bulb Economizer:
\$7742 - \$6661 = \$1081

\$1081/\$7742 = 0.139 = 13.9%

Total Savings With Ideal Economizer:
\$7742 - \$6565 = \$1177

\$1177/\$7742 = 0.152 = 15.2%

Table 2.2

Deck Costs With Fixed Hot Deck Temperature

(Setpoint = 88 F)

Costs in Dollars for 24 hr/day, 7 day/week Operation									
	On Minimum OA			With Economizer					
				Dry Bulb			Ideal		
	HD	CD	Tot	HD	CD	Tot	HD	CD	Tot
Jan	474	134	608	510	71	581	510	71	581
Feb	384	145	529	425	74	499	425	74	499
Mar	312	293	605	410	86	496	410	86	496
Apr	172	536	708	312	125	437	311	125	436
May	98	836	934	213	359	572	212	356	568
Jun	43	1177	1220	98	838	936	97	839	936
Jul	30	1391	1421	65	1246	1311	63	1195	1258
Aug	38	1319	1357	85	1116	1201	83	1070	1153
Sep	70	996	1066	156	609	765	154	608	762
Oct	149	630	779	293	162	455	293	161	454
Nov	260	342	602	381	64	445	380	65	445
Dec	394	195	589	453	81	533	453	81	534
TOT	2424	7994	10418	3401	4831	8231	3391	4731	8122

Total Savings With Dry Bulb Economizer:
$10418 - $8231 = $2187

$2187/$10418 = 0.21 = 21%

Total Savings With Ideal Economizer:
$10418 - $8122 = $2296

$1177/$7742 = 0.22 = 22%

The figures in Tables 2.1 and 2.2 clearly demonstrate that an economizer cycle can provide substantial savings in this climate. They also show a dry bulb economizer can provide most of the savings which could be achieved by an ideal economizer. Since an ideal economizer would require humidity sensors as well as a much more complicated optimization routine, it did not appear that the added potential for savings justified their use. The accuracy of at least some of the commercially available humidity sensors is open to question (15) and their use would require much additional time be spent in testing and calibration. Some interesting research into improved humidity sensors is currently being done by Caruso, Leidenfrost, Pearson, and DeWitt at Purdue (to be published in the proceedings of the 22nd ASME-AIChe National Heat Transfer Conference, August 1984), but their work is not directed toward HVAC applications. The probe they have developed shows considerable promise and may be suitable for HVAC systems, but it has not been tested under the "install and forget" standards required for this application. A leading controls manufacturer has stated that the difference in savings between an enthalpy controller and an ideal dry bulb economizer is minimal (16), a statement which is supported by these simulations. The most significant error introduced by not sensing humidity is in the cold deck cost calculations. Condensation in the cold deck can under some conditions cause the actual cold deck cost to be much higher than that calculated by the dry bulb optimization routine. This error does not affect the control decision; however, as program Pdeck showed that condensation does not become significant until the cooling load is so much greater than the heating load that maximum use of outside air is required anyway. Thus the dry bulb economizer will make the correct decision, but it will underestimate the savings resulting from this decision.

Another interesting conclusion which can be drawn from the tables is that the use of some sort of hot deck reset is very desirable in a dual duct system. Since the flow of air into each room is essentially constant, when the hot deck temperature increases the room dampers will call for a greater flow of cold air to temper the hot air. Thus increasing the hot deck temperature will increase cold deck costs as well as hot deck costs. Obviously a similar conclusion could be drawn for the cold deck temperature. One method of controlling these temperatures which would appear to be particularly attractive is that of zone reset. This type of control scheme lowers the hot deck temperature until the coldest room is calling for full flow from the hot deck, and raises the cold deck temperature until the warmest room is calling for full flow from the cold deck. The fact that a malfunctioning hot deck reset control can easily increase annual costs by 35%

also demonstrates the fact that tremendous savings can often be achieved just by repairing an existing system so that it operates as designed.

HARDWARE

As mentioned in Chapter 1, the hardware used in this research was selected primarily for its availability rather than its suitability for a long term control application. (Providing, of course, that it was accurate enough for use in a research project.) No attempt was made to obtain optimum use of the hardware or to construct a control system which would meet the durability and maintenance standards required of operational HVAC controls. In some cases commercially available HVAC components were used and in other cases laboratory or specially constructed equipment was selected. Fortunately these separate elements worked well together and the entire control system functioned very well throughout the test. The microprocessor control system together with the existing pneumatic controls as installed on HVAC unit ACP-7 are shown in Figures 3.1 through 3.3. A description of the microprocessor control elements (with more detailed photographs) follows.

3.1 Microprocessors

The microprocessor used in this experiment was a Texas Instruments model TI-9980A installed in a Texas

Figure 3.1

System ACP-7

1. Return Air Duct
2. Mixed Air Section
3. Supply Fan
4. Hot Deck
5. Cold Deck
6. Hot Air Duct
7. Cold Air Duct
8. Control Panel

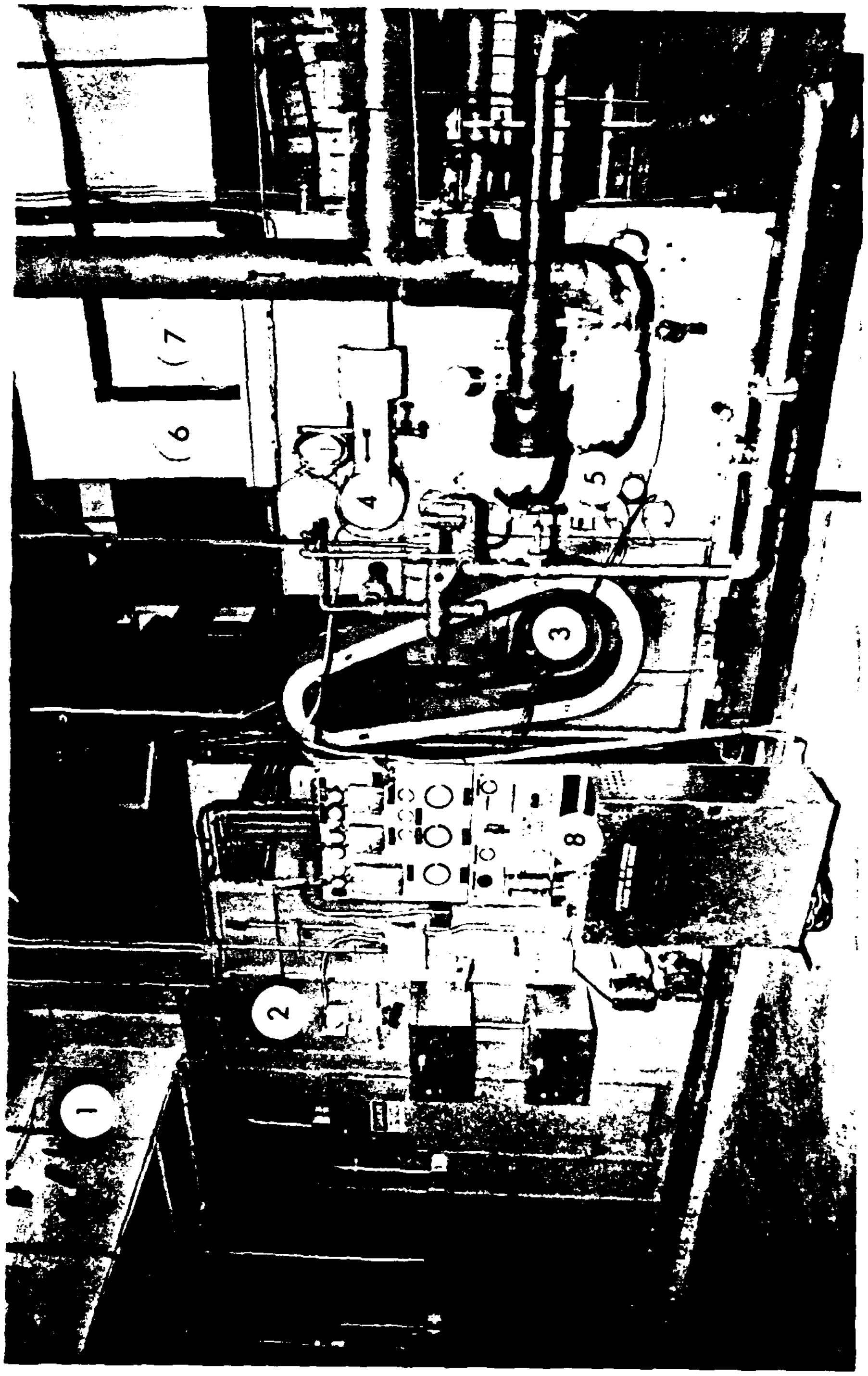


Figure 3.1
System ACP-7

Figure 3.2

Control Panel

1. Pneumatic Deck Controllers
2. Microprocessor/Pneumatic Interface Panel
3. Microprocessor Controls

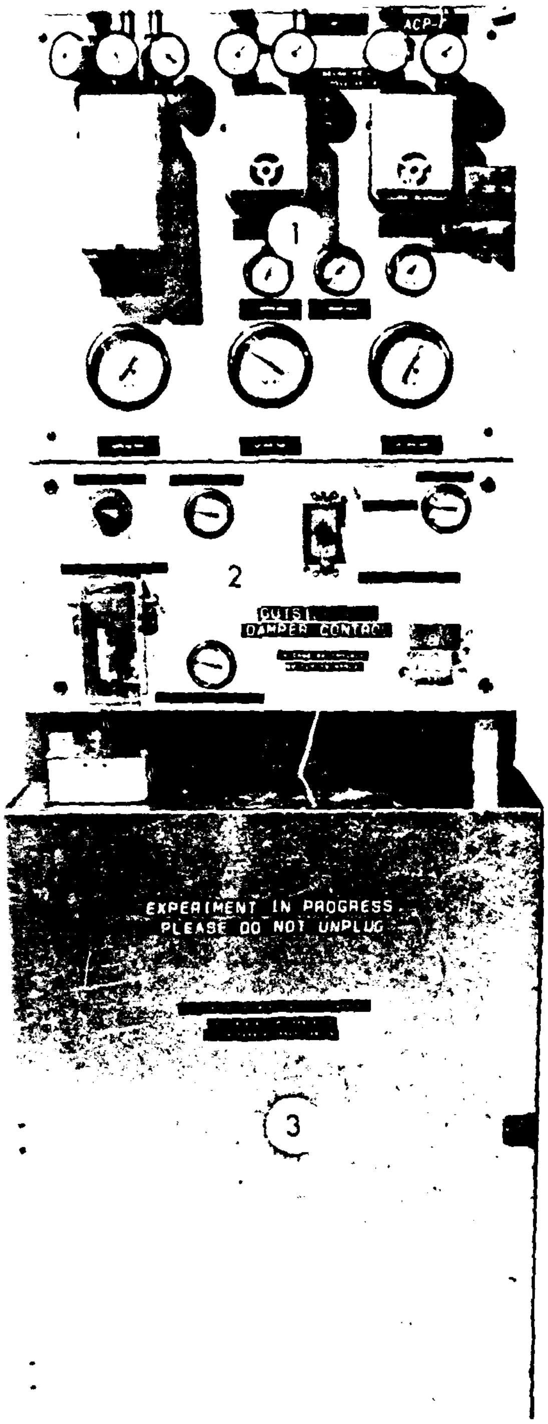


Figure 3.2

Control Panel

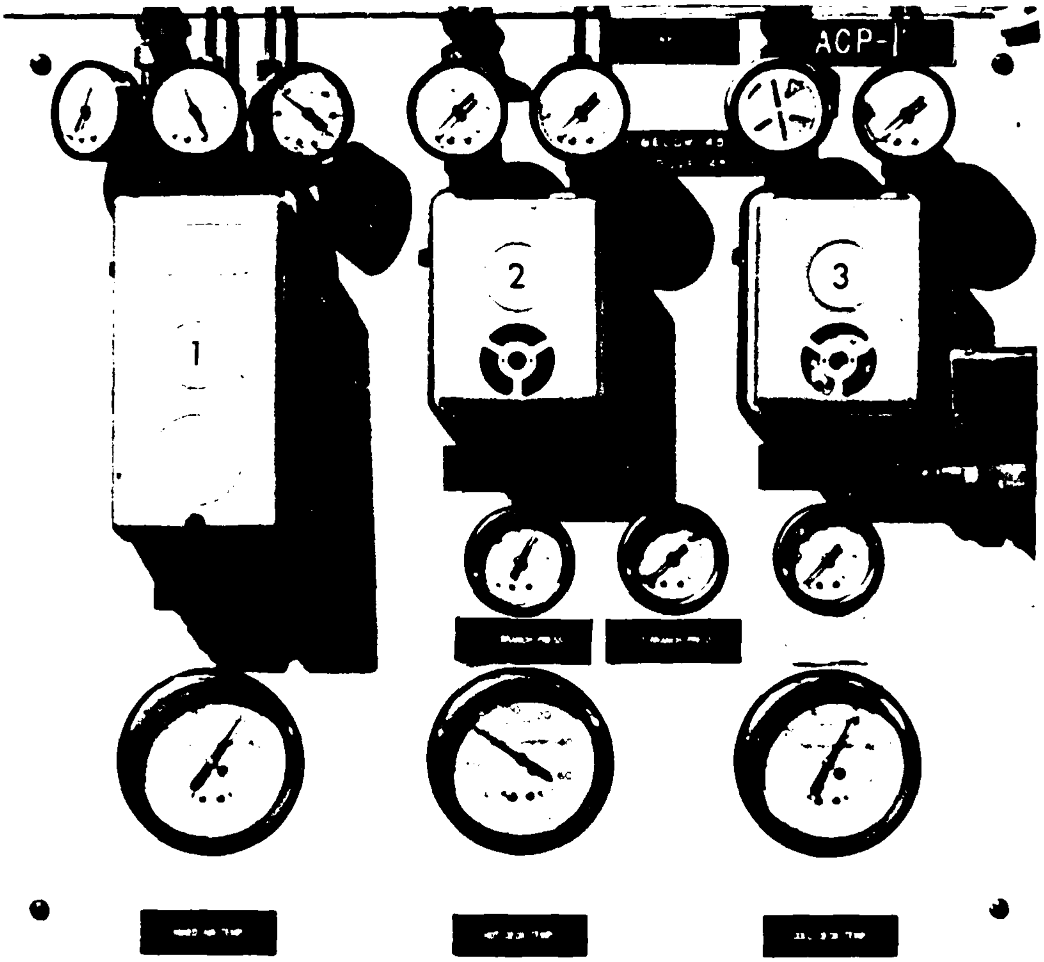


Figure 3.3

Pneumatic Deck Controllers

1. Hot Deck Controller
2. Mixed Air (Economizer) Controller (Disabled)
3. Cold Deck Controller

Instruments TM 990/189 microcomputer. (The microcomputer contains the RAM and ROM memory, input and output ports, keyboard, and other peripherals as well as the main microprocessor chip. For convenience, the terms "microprocessor" and "microcomputer" are used interchangeably in this thesis.) This 16 bit microprocessor runs on a 2 MHz clock and will handle both internally and externally generated interrupts. The memory options used in this experiment allowed for 2048 bytes of Random Access Memory (RAM) and 4096 bytes of Read Only Memory (ROM). The microprocessor is capable of accessing up to 16,384 bytes of RAM if an off-board expansion is used, but that was not required for this experiment. Similarly the on-board ROM capacity could have been increased to 6144 bytes to allow the control program to be indelibly programmed, but since the control program was used for experimental purposes only this was not done. The control program itself occupied approximately 1500 bytes of RAM and could easily have been implemented by a single microcomputer; however, the Analog to Digital (A/D) converters used to read the sensors worked more conveniently with two microcomputers so a dual microcomputer scheme was adopted. In this scheme one microcomputer, designated Micro 1, read all temperature sensors and performed the control calculations while the other microcomputer, Micro 2, read the velocity sensors and periodically logged data.

Communication between the two units was done at 300 baud via on-board serial communication ports. The communication, data logging, data processing, operating system overhead, and other non-control functions occupied approximately 1400 bytes of RAM, leaving 1200 bytes available for data storage. Most of these functions would not be required in an operational controller, but they were useful in this experimental application. Both microcomputers were equipped with a tape recorder interface which allowed programs to be loaded from or dumped to a portable tape recorder. This feature was used extensively to program the computers and to collect data from them. The microcomputers used together with the A/D converters, power supplies, and sensor electronics are shown in Figure 3.4.

3.2 Analog to Digital Converters

The A/D boards used in this experiment were designed by Professors Fearnot and Citron at Purdue for use with the TI microprocessor in a student laboratory. They are based upon a Burr Brown SDM857JG A/D chip, and the board was deliberately laid out in a "breadboard" type fashion to allow students to see the individual components. Dip switches allow the boards to be used as 8, 10, or 12 bit converters with single ended or differential inputs, and allow for various input voltages and internal gains. In this experiment both boards were used at the full 12 bit

Figure 3.4

Microprocessor Control Panel

1. Micro 1 (Master Control Microprocessor)
2. Micro 2 (Submaster Microprocessor)
3. A/D Convertor For Micro 1 (Temperature Sensors)
4. A/D Convertor For Micro 2 (Velocity Sensors)
5. Wheatstone Bridge Circuits For Temperature Sensors
6. Power Supply For Temperature Sensors
7. Power Supply And Circuitry For Velocity Sensors
8. Power Supply For Microprocessor

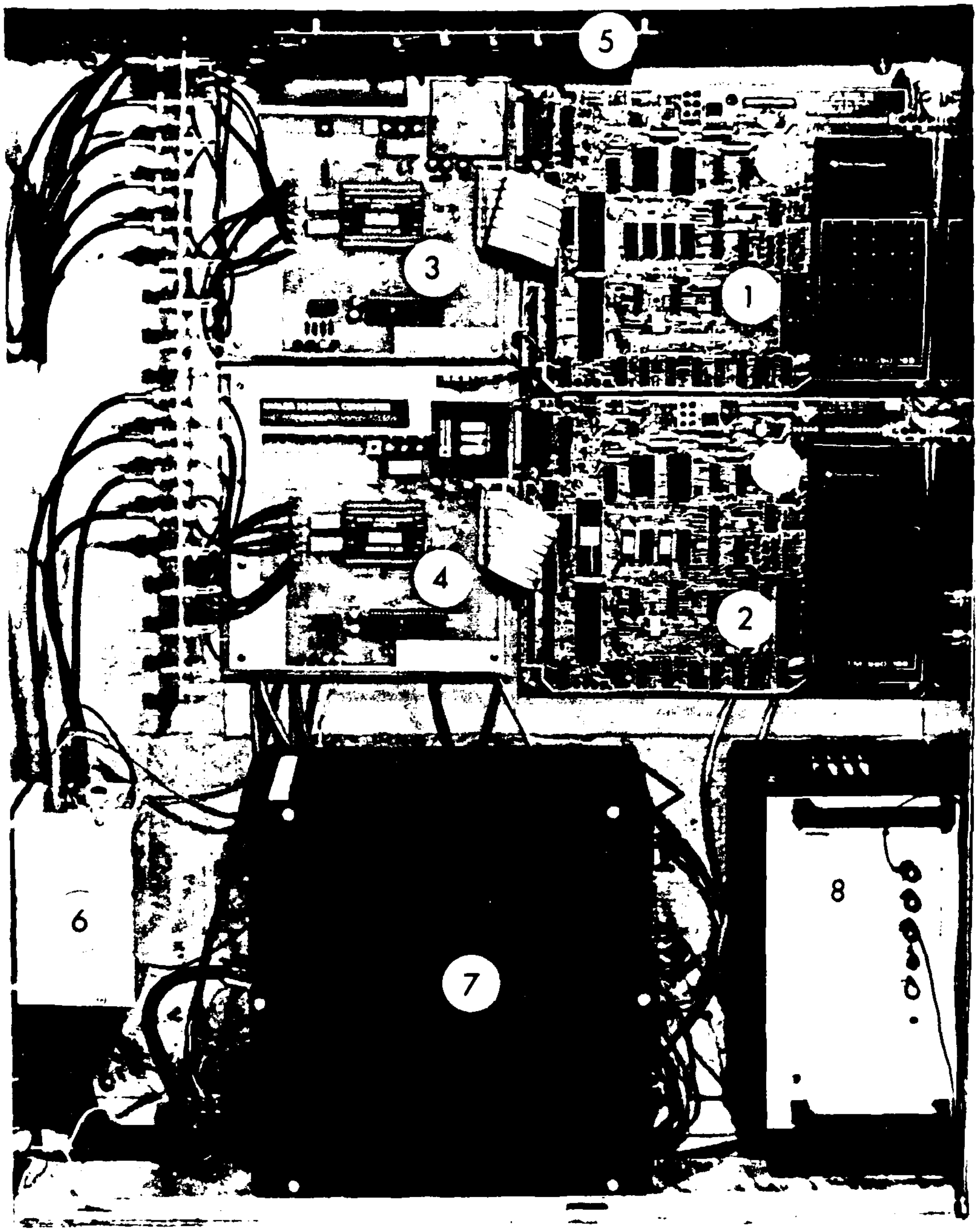


Figure 3.4

Microprocessor Control Panel

resolution and differential inputs were used throughout. The A/D board used to read the temperature sensors was set for ± 5 volt input to the A/D chip with a 20x amplifier between the board inputs and the chip, allowing for ± 0.25 volt sensor inputs. The A/D board used to read the velocity sensors was set for a 0-10 volt input to the A/D chip with a 2x amplifier between the board inputs and the chip, allowing for 0-5 volt sensor inputs.

3.3 Temperature Sensors

The temperature sensors used were Johnson Controls TE-1100 series nickel wire thermistors. These sensors are standard HVAC elements and are commonly used with DDC units. Single point sensors were used in the outside and return air ducts, while 16 ft. long averaging sensors were used in the hot deck, cold deck, and mixed air sections. These two types of sensors are shown in Figure 3.5. The sensors have a nominal resistance of 1000 ohms at 70 F (± 1%) and will increase their resistance by approximately 3 ohms for every 1 degree F. increase in temperature. A Wheatstone bridge circuit powered by a dedicated 5 volt power supply was used to measure this resistance change. (The 5 volt power supply was selected after bench tests showed higher voltages could cause self heating problems. The 5 volt power supply caused self heating of approximately 0.1 F in still room air, an increase which is insignificant in HVAC

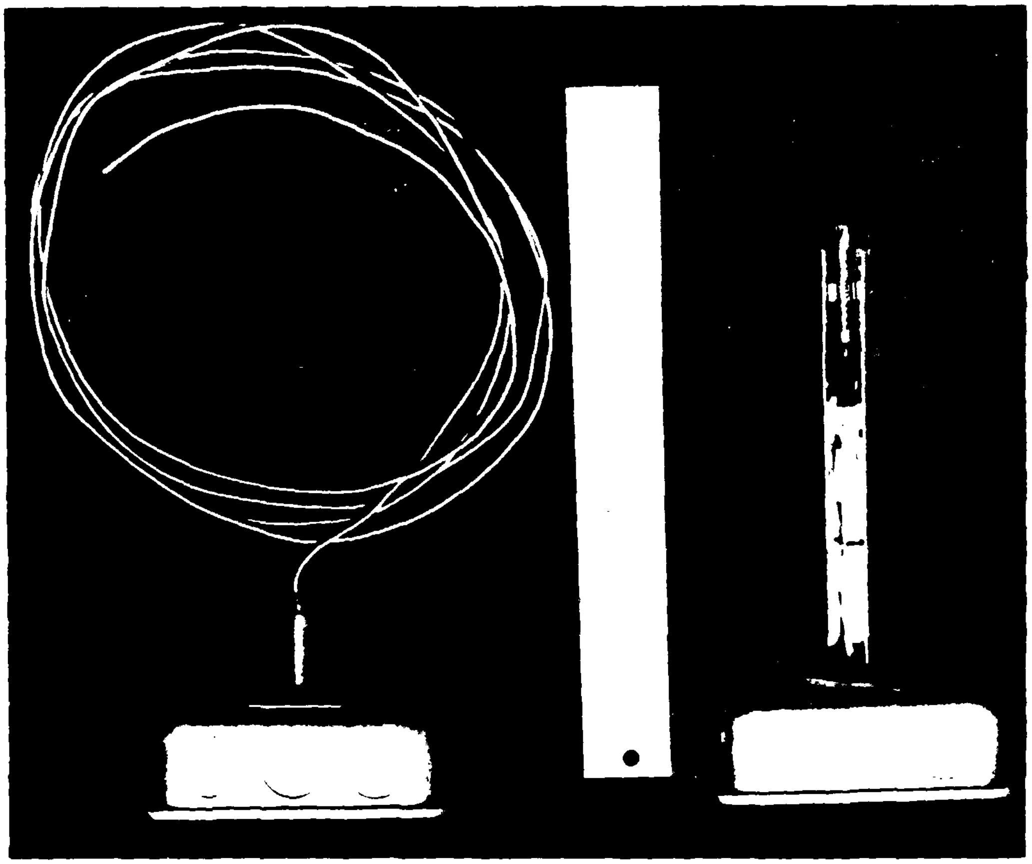


Figure 3.5

Temperature Sensors

Averaging Sensor (left) and Point Sensor (right) work.) Based upon the results of program Pdeck, the outside air sensor was expected to experience the widest temperature variation, and this would normally remain within a range of -20 F to 100 F The midpoint of 40 F was therefore chosen as the balance temperature for the bridge circuits. This required bridge resistors of 914 ohms (based upon Johnson literature), the resistors actually used averaged 920 ohms and gave a balance temperature of 40.8 F. Johnson's literature predicted a resistance of 803 ohm at 0 F and a resistance change of 2.8667 ohm per degree, therefore:

$$\text{Temp} = (R_{sensor} - 803)/2.8667 \qquad 3.1$$

A diagram of a Wheatstone Bridge circuit is given as Figure 3.6. The standard equation which predicts the output from this circuit (ignoring for the moment the small calibration potentiometer) is:

$$V_{out} = V_{ref} \left| \frac{R_1 R_4 - R_2 R_3}{(R_1 + R_2)(R_3 + R_4)} \right| \qquad 3.2$$

where:

V_{out} = bridge output voltage

V_{ref} = reference voltage used as bridge input

R_1 through R_4 = Resistors used in each leg of the bridge

Selecting R_1 as the resistance of the temperature sensor, equation 3.2 can be solved for R_1 as:

$$R_1 = -R_2 \frac{(R_3 + R_4)\ V_{out}/V_{ref} + R_3}{(R_3 + R_4)\ V_{out}/V_{ref} - R_4} \qquad 3.3$$

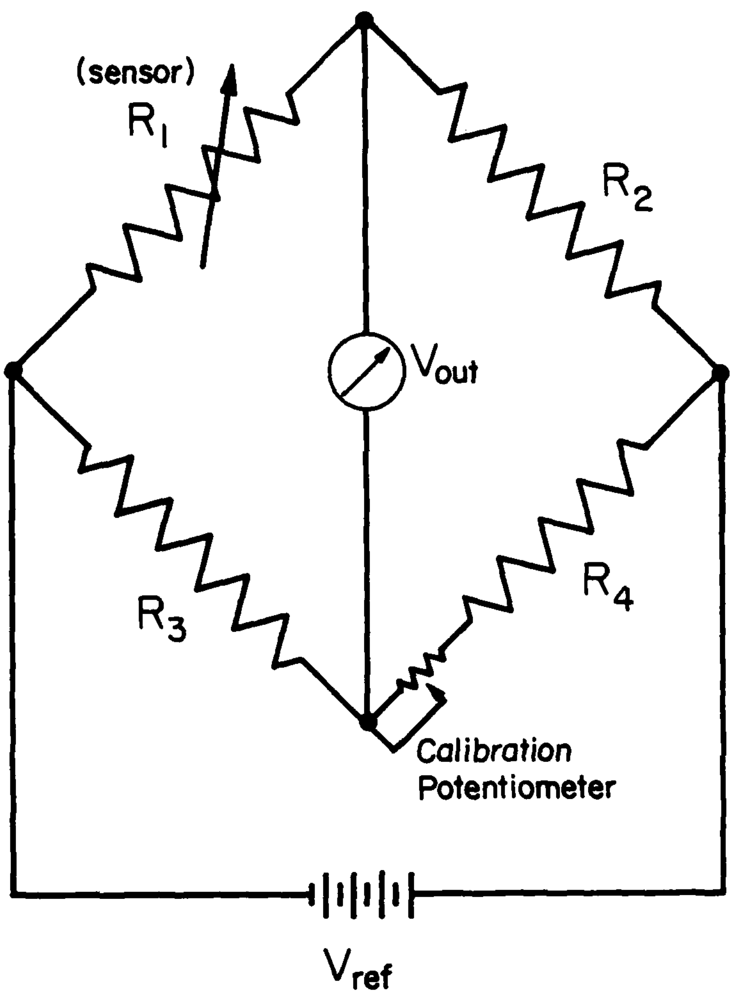


Figure 3.6

Wheatstone Bridge

Using the known values of V_{ref} = 5 V and R_2 through R_4 = 920 ohm Equation 3.3 can be written as:

$$R_1 = -920 \frac{368\ V_{out} + 920}{368\ V_{out} - 920} \qquad 3.3a$$

Combining equation 3.3a with equation 3.1 yields:

$$Temp = \frac{1}{2.8667} \left| \frac{-920\ (368\ V_{out} + 920\ )}{(\ 368\ V_{out} - 920\ )} - 803 \right| \qquad 3.4$$

This was the basic equation used to convert the bridge voltage V_{out} into a temperature reading. The leg of the bridge circuit opposite the sensor contained a 15 turn potentiometer which was used to calibrate the null point, and the microprocessor software contained a span constant which was used to change the slope of the curve. This had the effect of altering equation 3.2 to:

$$V_{out} = V_{ref} \left| \frac{R_1 R^* - R_2 R_3}{(R_1 + R_2)(R_3 + R^*)} \right| \frac{SPAN}{1000} \qquad 3.2a$$

where

$$R^* = R_4 + R_{potentiometer}$$

and

SPAN = Span adjustment (in microprocessor software)

The potentiometer settings and span constants were unique to each sensor, they compensated for small variations in the five individual sensors and bridge circuits. This allowed one microprocessor subroutine to read all five sensors. Obviously a span adjustment of 1000 would have no effect on the temperature readings, spans greater or less than 1000 change the slope of the curve.

3.3.1 Modifications for use in a microprocessor Equation 3.4 is the basic equation which describes temperature as a function of voltage. To use this equation in a microprocessor it had to be modified slightly. To begin with, the input to the microprocessor was not a voltage but was instead the digital output of an A/D converter. The A/D converter used was a 12 bit converter which meant that the largest number it could output was $2^{12} - 1$ or 4095. Since the inputs could be positive or negative; however, one bit had to be reserved to indicate the sign of the number so only eleven bits were left to indicate the magnitude. The largest number which could be input to the microprocessor was therefor $2^{11} - 1$ or 2047. As mentioned previously the A/D board was set for a $\pm$ 0.25 volt input, which meant that a 0.25 volt input resulted in a digital output of 2047, so:

$$D_{out} = V_{out} \frac{2047}{0.25}$$

or

$$V_{out} = \frac{D_{out}}{8188} \qquad 3.5$$

Substituting this into equation 3.4 yields

$$Temp = \frac{-320.929\ (0.0449\ D_{out} + 920)}{0.0449 D_{out} - 920} - 280.11 \qquad 3.6$$

Equation 3.6 could not be implemented by the microprocessor as written because the microprocessor handles integer numbers only and will ignore all decimals. In addition, it was desirable to multiply the right hand side of equation 3.6 by 10 so that the number calculated by the microprocessor would be 10 times the actual temperature. This allowed subsequent calculations to be performed to an accuracy of 1/10th of a degree, i.e. a temperature of 40.9 F could be expressed as the integer 409. The equation actually used by the microprocessor to convert the A/D output to a temperature, therefore, was:

$$\text{Temp} \times 10 = \frac{3209\ (D_{out} + 20470)}{20470 - D_{out}} - 2801 \qquad 3.7$$

3.3.2 <u>Notes on accuracy</u> The largest number which can be stored in a single register of a 16 bit microprocessor (reserving one bit to indicate sign) is $2^{15} - 1$ or 32,767. The TI microprocessor normally works with single registers; however, when two single registers are multiplied it stores the output in two successive registers allowing for numbers slightly in excess of 2×10^9. Similarly when the TI divides two numbers, it starts with the dividend stored in two registers and places the answer in a single register. This means that greater accuracy can be obtained if multiplications are paired with divisions whenever possible. The software implementing equation 3.6 was written to take advantage of this fact, as was much of the software written for this experiment. This software was tested with the highest and the lowest temperatures expected and no problems with overflow or underflow were encountered. The resolution of these sensors can easily be found by substituting $D_{out} = 0$ and $D_{out} = 1$ into equation 3.7. For $D_{out} = 0$ this yields Temp = 40.8 and for $D_{out} = 1$ this yields Temp = 40.8314. Thus the A/D converter can indicate a change of 0.03 F, although of course the microprocessor only records changes of 0.1 F.

3.3.3 Calibration To calibrate the temperature sensors the span adjustment was initially set to 1000, the sensors were immersed in a water bath at approximately 40.8 F, and the potentiometer was adjusted until the temperature reading agreed with a laboratory thermometer immersed in the same water bath. The sensors were then immersed in a hot water bath and readings were taken while the temperature was slowly lowered by adding small amounts of ice. The temperature was allowed to stablize before the reading was taken on each step. The calibration was checked in this manner over a range of approximately 90 F to 32 F. Small changes were made to the span and zero adjustments as needed and the calibration procedure was repeated until a reasonably accurate calibration was achieved. The calibration curve for the outdoor air sensor is given as Figure 3.7, other sensors had similar curves. The calibration of each sensor was adjusted to give maximum accuracy

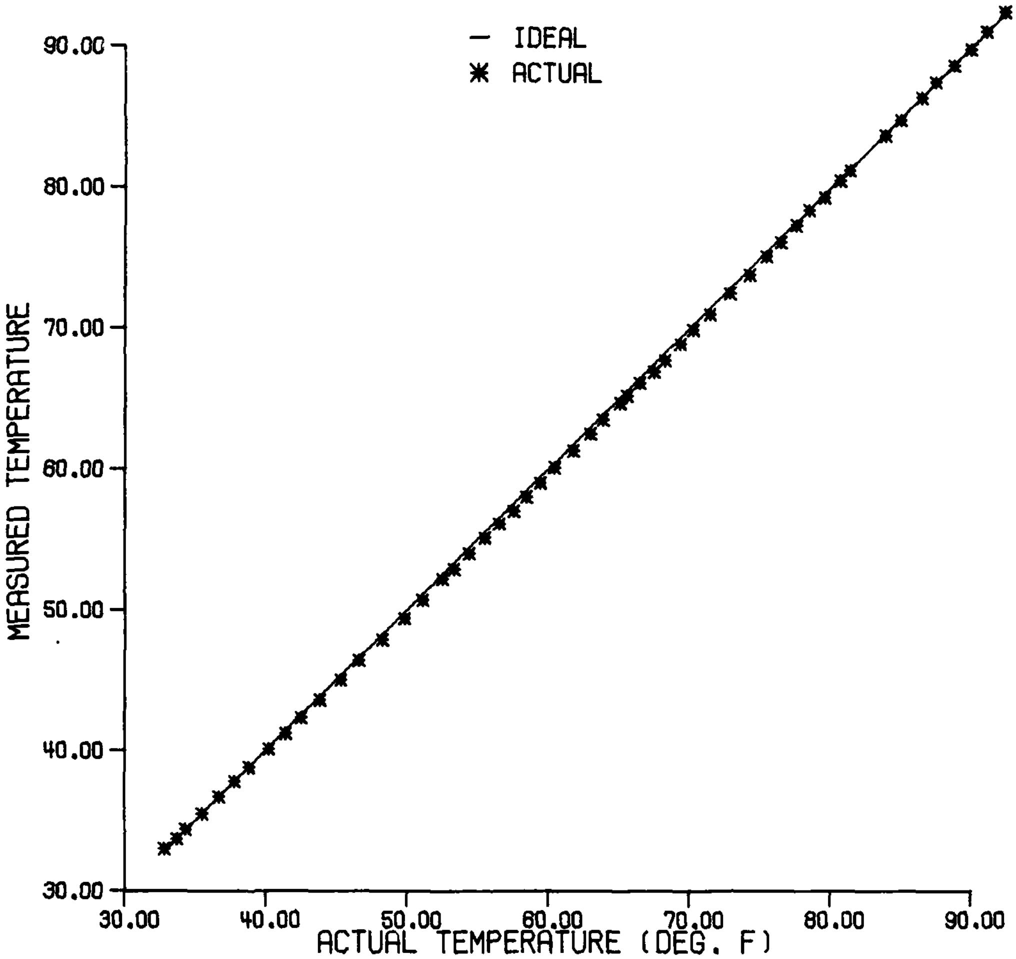


Figure 3.7

Temperature Sensor Calibration

in the temperature range where it would most often be operating.

After the sensors were installed in the ductwork the calibration was again checked with a laboratory thermometer and the potentiometers were adjusted as required to compensate for the resistance of the wires leading to the sensors. This calibration was checked weekly throughout the experiment. Only the zero-adjust potentiometers were changed during these in-place calibrations; the span adjustments were never altered.

3.4 Flow Sensors

To calculate the coil loads of system ACP-7 it was necessary to know the temperature differences across the coils and the flow rates through them. Either the air side or the water (steam) side of the coils could have been monitored, in this experiment measurements were taken on the air side because it was felt the sensors would be less expensive. The temperature sensors have already been described, the flow sensors used were heated thermistor probes measuring the centerline velocity in the ducts. These probes were built specially for this project using circuitry developed by ITT. (17) One of the probes used is shown in Figure 3.8. The inset in this photograph shows a close-up of the flow sensing and temperature compensating thermistors mounted on the tip of the probe.

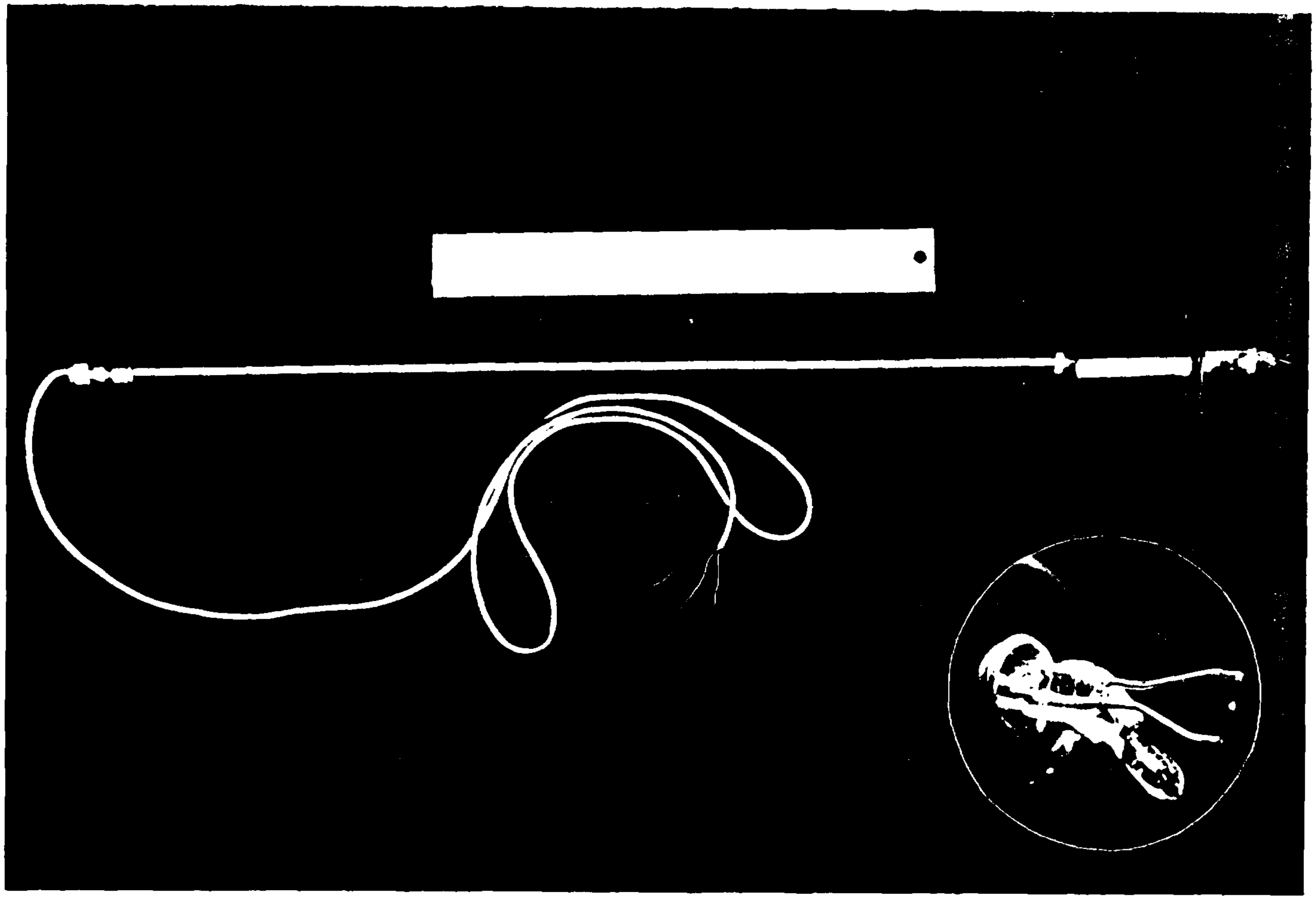


Figure 3.8

Velocity Sensor

The probe mounting was designed to allow the tip to be withdrawn into a protective tube during transport and installation, and then extended to the duct centerline during use. (The tip is shown partly extended in Figure 3.8) Figure 3.9 shows an installed velocity sensor with the probe fully extended into the duct. As with the temperature sensors, variations between the individual sensors were compensated for using a potentiometer for zero adjust and a software constant for span adjustment.

3.4.1 Calibration: The calibration of the velocity sensors was done in a wind tunnel maintained by the Testing and Balancing section of Purdue Physical Plant. This tunnel was designed for this type of use and was particularly well suited to the low velocities encountered in HVAC ducts. The initial calibration was done using a digital voltmeter to measure the sensor output at various flow rates. Because the duct flow was turbulent and the response time of the thermistor was extremely rapid, a fairly wide range of voltages was output for any given average velocity (as measured by a flow nozzle and manometer). To facilitate calculations on the microprocessor the curve measured in this test was modeled as two straight line segments. These lines were described by:

$$V = (E - 1.56)(944) \qquad \text{for } E > 2 \text{ v} \qquad 3.7a$$

and

Figure 3.9
Velocity Sensor Installed In Duct

$$V = (E/2)\ 400 \qquad \text{for } E < 2\text{ v} \qquad 3.7b$$

where:

V = Velocity (feet per minute)

and

E = Sensor output voltage

The measured voltages and the modeling curves are shown in Figure 3.10. This figure only shows the voltage readings which occurred most often. Since the readings were taken to model the average velocity curve the extreme fluctuations of voltage were not recorded.

As with the temperature readings, equations 3.7 needed to be modified slightly for use in a microprocessor. Since the A/D board was set to give a maximum output of 2047 for a 5 volt input:

$$E = D_{out}\ (5/2047) = D_{out}/409.4 \qquad 3.8$$

Equation 3.8 shows that the 2 volt breakpoint in equations 3.7 corresponds to a digital output of 819. Using this as a breakpoint and substituting equation 3.8 into 3.7 yields:

$$V = \frac{231\ D_{out}}{100} - 1473 \qquad \text{for } D_{out} > 819 \qquad 3.9a$$

and

$$V = \frac{400\ D_{out}}{819} \qquad \text{for } D_{out} < 819 \qquad 3.9b$$

Equation 3.9 was the basic equation used to convert the

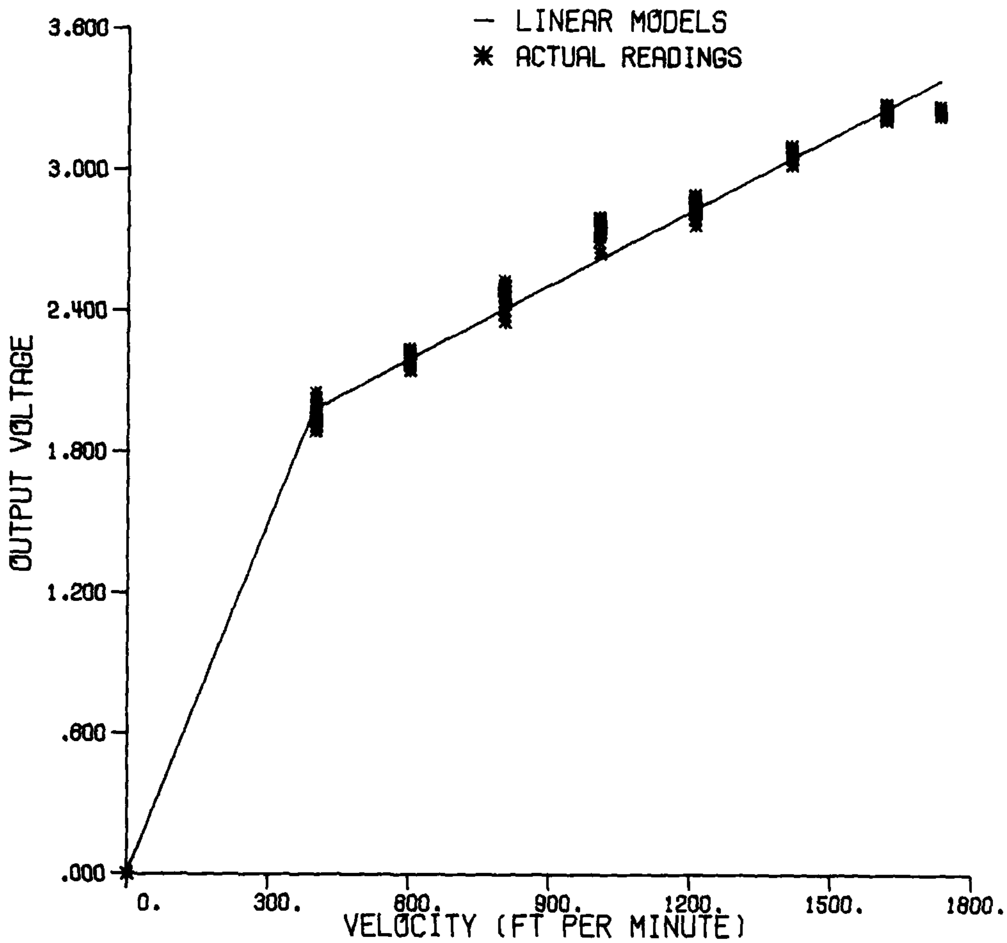


Figure 3.10

Initial Velocity Sensor Calibration

A/D output to a velocity reading. Initial tests with sensors using this equation yielded rapidly fluctuating velocities due to turbulence in the ducts. To help smooth out the effects of this turbulence the A/D board was read 256 times each time a velocity reading was required. The average of these 256 instantaneous readings was then converted to an average velocity reading using equations 3.9. The software span adjustment was multiplied times the output of equation 3.9 so that:

$$V^* = V\ (SPAN/1000)$$

where

V^* = Corrected velocity for use in flow calculations

V = Velocity calculated by equation 3.9

SPAN = Software span adjustment

These equations were programmed into a microprocessor and each velocity sensor was individually calibrated in a wind tunnel. During the calibration process the probe was initially sealed against stray drafts, the zero adjust potentiometer was adjusted upward until a positive reading was achieved and then slowly backed off until a position was found where twenty consecutive readings produced a zero output. (The zero had to be found in this manner because the software rejected negative readings and set them equal to zero.) The probe was then placed in the wind tunnel and exposed to an 800 fpm flow. The software constant was adjusted until consecutive readings averaged 800 fpm. The calibration was then checked over a range of 0 to 1100 fpm (the predicted operating range in system ACP-7), and the entire process was repeated if necessary. A typical sensor calibration curve is given in Figure 3.11. Note that even though 256 readings were averaged, there is still a range of roughly $\pm$ 10% present at most velocities. This is most probably caused by slow oscillations in the turbulent flow. The microprocessor can take 256 readings in a fraction of a second so slow variations in the flow will not be averaged out. Figure 3.11 shows 30 average velocity readings (256 instantaneous readings per average reading) taken at each interval of 100 fpm actual velocity. These readings were taken approximately 5 seconds apart. Unlike Figure 3.10, the extremes were not omitted from Figure 3.11 and all readings are plotted. The average of the 30 readings taken at each step compares very well with the actual velocity, except in the vicinity of the 400 fpm breakpoint. This may be at least partly explained by the fact that fluctuations which fall below this point are converted into velocity using a curve with a steeper slope, hence the average is lower than if the same slope was used on both sides of this point. The fact that an individual velocity reading could differ from the actual velocity by as much as 10% (or slightly more in the vicinity of 400 fpm) did not cause any significant

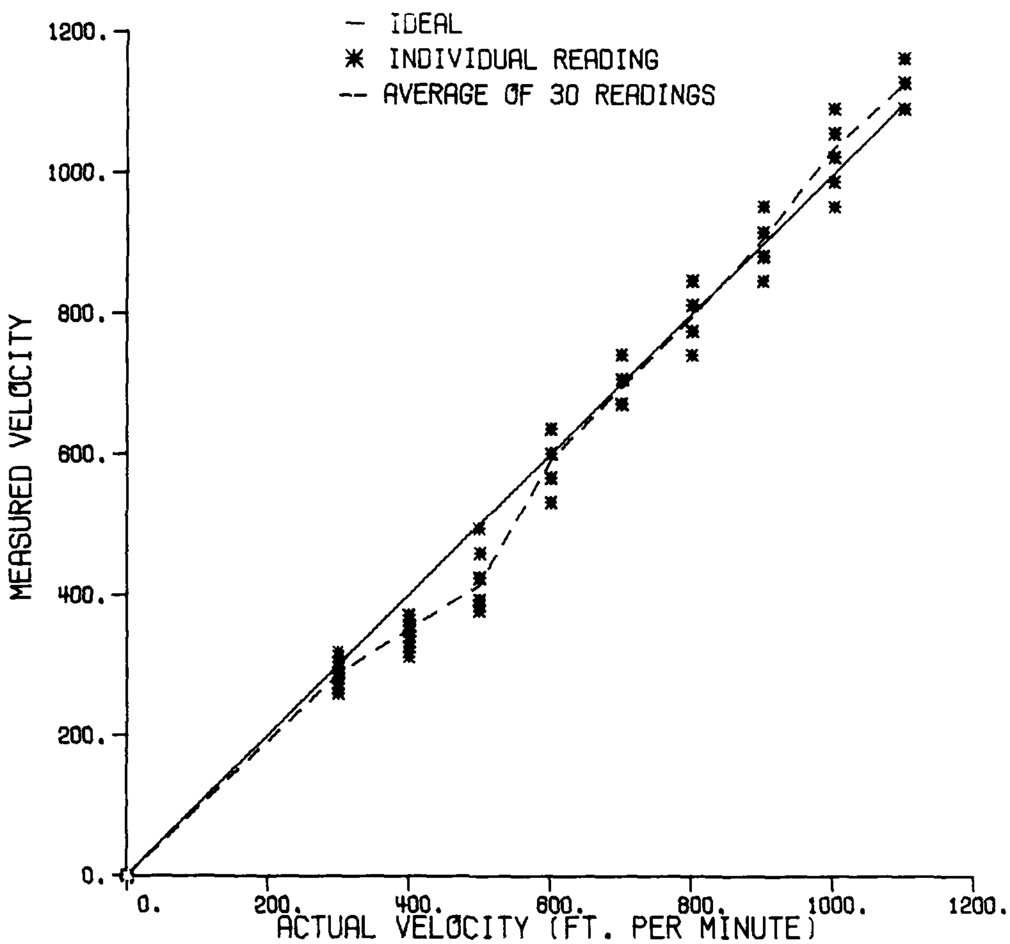


Figure 3.11

Final Velocity Sensor Calibration

problems in the optimization routine. The results of program Pdeck showed that most of the time system ACP-7 would be operating in either a clearly defined heating mode (requiring minimum OA) or a clearly defined cooling mode (requiring a mixed air temperature equal to that of the cold deck) and would very seldom be operating in a range where a 10% error in the flow readings would change the control decision.

Once the flow sensors were mounted in the ductwork, the zero adjust was set to compensate for the longer leads. A hot wire anemometer was used to check the accuracy of the velocity readings on a weekly basis, and slight adjustments to the calibration were made as required. Unlike the temperature sensor calibration, these adjustments were made to the software span factor. Since a zero velocity could be produced by turning off the fan and withdrawing the sensors into their protective tubes, the potentiometer zero adjustment could be set exactly and did not need to be altered during the experiment.

3.4.2 Conversion of centerline velocity to flow rate To calculate the airflow through the ductwork in system ACP-7 it was necessary to multiply the average velocity by the cross sectional area of the duct. The velocity profiles for fully developed flow in round ducts have been studied extensively and several methods for determining the average velocity from a local velocity are known, but HVAC systems typically use square or rectangular ducts. There does not appear to be a great deal of published material on velocity profiles in non-circular ductwork, but what little there is points to the idea that the ratio of $V_{average}/V_{centerline}$ will remain fairly constant throughout a wide range of flow rates. The flow in HVAC ducts tends to be highly turbulent with a relatively flat velocity profile. The boundary effects are limited to a region close to the duct walls, and so this region has only a slight effect on the total flow rate. Miller (18) studied the flow through flat oval ducts (system ACP-7 used rectangular ducts) and showed that the ratio of $V_{average}/V_{centerline}$ varied between 0.88 and 0.98 depending on the aspect ratio of the duct and the Reynold's number of the flow. Ahmed and Brundrett (19) studied flow in the entrance region of a square duct and found that $V_{average}/V_{centerline}$ varied from 0.77 to 0.88 in their test. Measurements taken by the testing and balancing section of the Purdue Physical Plant on system ACP-7 showed this ratio ranged between 0.88 to 0.95. Based upon these various sources the value used in this experiment was:

$$V_{average} = 0.9\ V_{centerline} \qquad 3.10$$

Obviously this figure could be refined at some future time if additional research is done in this area. The calibration readings taken throughout the project showed that a flat velocity profile did in fact exist in the ducts. The boundary layer effects tend to be most noticeable at lower flow rates, so the assumption of a constant $V_{average}/V_{centerline}$ ratio may not be valid for low flow rates. Since the total air conditioning system basically operates at a constant flow rate a low flow rate in, say, the hot deck means the cold deck is experiencing a high flow rate. Thus the assumption of a flat velocity profile may not be valid for the hot deck flow calculations, but this will not affect the control decision since the much greater flow in the cold deck will require economizer operation regardless of any errors in the hot deck calculations. Similarly if a flat profile exists in both ducts but the assumed value of 0.9 is in error it will affect the cost calculations but will not affect the control decision since the same error will be introduced into both the hot deck and the cold deck calculations.

The location where these sensors were mounted had an important bearing on their performance, since the flow in the duct had to be fairly well established for the assumptions regarding centerline velocity to be valid. Both the hot and cold decks of system ACP-7 split into two ducts apiece immediately downstream of the coils. There was not sufficient room to allow the flow to become fully developed before this split occurred, so it was impossible to use one flow sensor in the hot deck and one in the cold deck. The two hot air ducts and two cold air ducts emerging from this split make a right angle bend and then run straight for approximately 20 feet before branching off in various directions. The flow sensors were placed near the end of this 20 foot run. Four sensors were used, one in each of the four ducts, and the microprocessor software computed the individual flows through each duct before adding them together into the total hot and cold deck flowrates.

3.5 Interface With Existing Controls

A schematic of the interface between the microprocessor and the existing controls is given in the appendix (Figure A.5) and a photograph of the interface board is shown in Figure 3.12. Basically this interface consisted of an electric to pneumatic (E/P) transducer controlled by the microprocessor. This transducer in turn controlled the air pressure to pneumatic damper motors, thereby controlling the percentage of outdoor air which is admitted into the system. The microprocessor did not power this transducer directly, as it required more current than the output chips could safely furnish. Instead two pins of a parallel output port on the microprocessor were used to control relays which switched ± 12 volts to the transducer

Figure 3.12

Microprocessor/Pneumatic Interface Panel

1. Minimum Position Adjustment
2. Electric/Pneumatic Transducer
3. Microprocessor/Manual Control Selector Switch
4. Electric/Pneumatic Relay

Figure 3.12
Microprocessor/Pneumatic Interface Panel

as required. A manual override was provided by a minimum position adjustment which could be used to send a fixed pressure to the dampers. A double pole single throw switch controlling an electric to pneumatic relay was used to select either manual or microprocessor control of the dampers. The second set of contacts on this switch was used to disable the microprocessor output relays when manual control was selected. This prevented the microprocessor from driving the E/P transducer to either extreme when it was not actually exercising control over the system. These controls were used to override the microprocessor control system during start-up and adjustment procedures, and were used on a daily basis to provide comparative data. Throughout the course of this experiment system ACP-7 was run on minimum OA for one day (using the manual adjust) and on microprocessor control the following day. The costs for operating in each mode were recorded and will be discussed in chapter 6. The interface components were standard HVAC control items, the E/P transducer was the only component which is of interest to the control strategy so it is the only item which will be described in detail.

The E/P transducer used in this experiment was a Johnson model EPT-101-1. Basically this transducer consists of a pneumatic regulator controlled by a reversible DC motor. The span and range of the pneumatic output was adjustable (within the 0-20 psi limit imposed by the 20 psi supply pressure) and was set to approximately 2-15 psi for this experiment. This allowed the microprocessor to exercise complete control over all dampers in the system. There was no spring return in this transducer, which meant that the output pressure remained constant in the absence of any electrical signal from the microprocessor. A + 12 volt signal from the microprocessor output relays would increase the output pressure and a -12 volt signal would drop the pressure. The degree to which the output pressure was changed was determined by the length of time the motor was driven. This is commonly referred to as "pulse width modulation" (PWM) control. The microprocessor measured the control error and calculated how much correction was needed, then it sent an appropriate control pulse to the transducer. The polarity of this pulse was determined by the sign of the error, and the duration was determined by the degree of correction needed. The advantage of this type of control scheme is that the system being controlled will remain stationary if the microprocessor "crashes" and ceases to operate. One interesting aspect of the transducer is that it serves as a mechanical integrator and the output pressure is the sum of all previous output pulses. This would be important if a theoretical evaluation of the controller gains (to be discussed in chapters 4 and 5) were attempted. In this experiment the gains were determined by a trial and error approach, as the non-linearities in the system would have made a theoretical evaluation very difficult.

MICROPROCESSOR SOFTWARE

As explained in the previous chapter, the design of the analog to digital converters required the use of two microcomputers in this experiment. The main control computer, hereafter referred to as "micro 1", read the temperature sensors, performed the control calculations, and transmitted control signals to the dampers. The software program executed by this computer was titled "Emaster", as it was the main economizer program. The other computer, called "micro 2", read the velocity sensors and logged data according to instructions from micro 1. The software for this computer was titled "Esub". Communication between the two computers was accomplished at 300 baud using standard ASCII characters. Existing RS-232-C serial communication ports on each computer were used for this communication, and "extended operation instructions" or "XOP´s" (subroutines written by the manufacturer) were used to control the actual receipt and transmission of signals. The protocol followed in this communication will be described later.

4.1 Program Emaster

A flow chart for the overall structure of program Emaster is given in Figure 4.1, and a copy of this program is included in the appendix. The format used in programming Texas Instruments microprocessors is:

label	operation code	comment

Using a line from program Emaster as an example:

```
ready   xop r5,13    wait for "on" signal from micro2
```

The label of this line is "ready", thus the microprocessor can be instructed to begin executing the section of the program which starts with this line by telling it to "jump to ready". The operation code is "xop r5,13". This is an assembly language instruction which tells the microprocessor to execute xop 13 (a "read" subroutine programmed in ROM) and to place the results in register number 5. Everything which follows this operation code is ignored by the microprocessor and is used to document the program. Additional comments are inserted throughout the program and are identified by an asterisk in the first column of the line. This tells the microprocessor to ignore that entire line. Further details of the programming language are beyond the scope of this thesis, several books on this

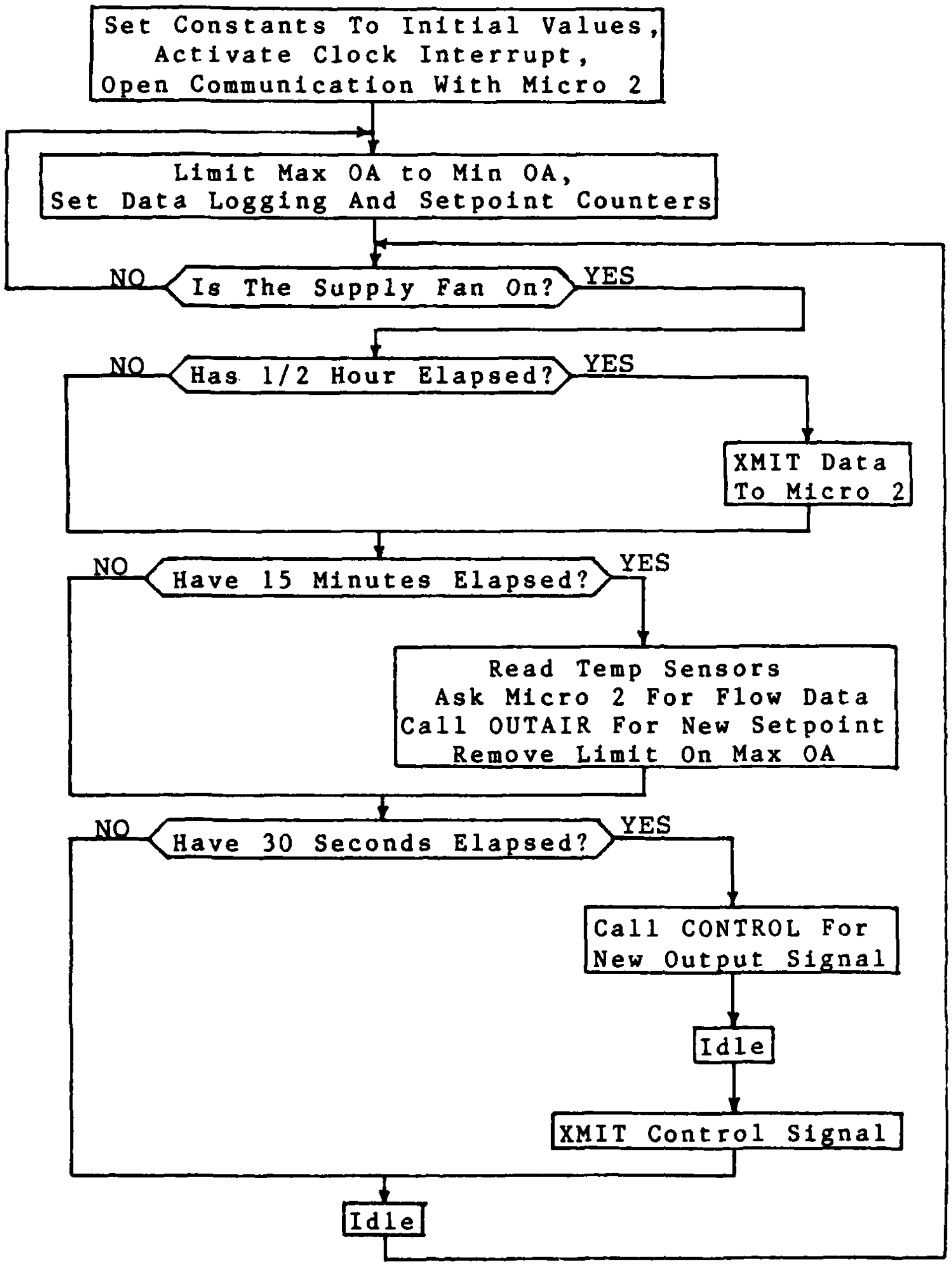


Figure 4.1

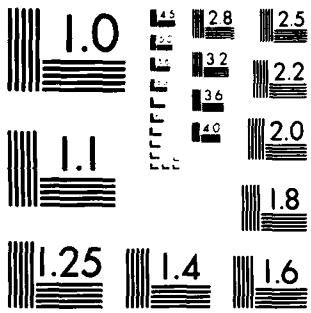
Program EMASTER (Main Program)

subject are published by Texas Instruments. (20-22) The program itself was written and edited on a high level computer (part of the Engineering Computer Network at Purdue). A cross-assembly program written by Texas Instruments and modified at Purdue was used to translate the assembly language program into a machine language version which could be loaded into the microprocessor.

4.1.1 Interrupts and Timing To provide stable control of system ACP-7, it was necessary to perform various tasks at equally spaced time intervals. Some of the considerations which went into choosing these intervals will be discussed in chapter 5, for now it will suffice to say that an output signal to the E/P transducer was generated every 30 seconds, the optimum percent outside air (and associated mixed air temperature) was calculated every 15 minutes, and data were transmitted to micro 2 every 1/2 hour. The timing of these actions was accomplished by counting interrupts. An interrupt is a signal generated from a source other than the software which causes the microprocessor to interrupt whatever calculations it is currently performing and begin executing a different section of its program. Oftentimes this section of the program resembles a subroutine, the difference is that a subroutine is called by the software while an interrupt is called by some other means. When an interrupt occurs the microprocessor stores all the data it needs to return to the calculations it was performing before the interrupt. In program Emaster an internal clock interrupt was used to control the timing of the program. The microcomputer contains a clock which can generate interrupts at intervals of up to 0.524 seconds, in this program it was set to cause an interrupt every 1/2 second. An "idle" statement at the end of the main program loop caused the microprocessor to halt its execution at that point and wait for this interrupt signal. Once the signal was received, the microprocessor would execute its "interrupt service routine" (the subroutine called by the interrupt) and then return to the main program. When the microprocessor returned to the main program it returned to the step after the "idle" instruction, and therefore executed the main loop again before returning to the idle step. The timing of various branches of the main program loop could thus be controlled by the fact that the loop itself was executed every 1/2 second. If a branch needed to be executed every 10 seconds, for example, a counting variable could be used to cause the branch to be bypassed during 19 passes through the loop and executed on the 20th pass The actual time it took to execute the main loop depended on which branches were selected, but the "idle" step guaranteed that the loop would only be executed once every 1/2 second. The "interrupt service routine" itself merely updated a variable which contained the current 24 hour

AD-A144 136 DEMAND CONTROLLED ECONOMIZER CYCLES: A DIRECT DIGITAL CONTROL SCHEME FOR..(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH S T TOM MAY 84 2/3

UNCLASSIFIED AFIT/CI/NR-84-33D F/G 13/1 NL



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963-A

clock time, so that the time at which various readings were taken could be recorded.

The timing control described in the preceding paragraph is based upon the assumption that the main program loop will take less than 1/2 second to execute. In general this was true, the loop could normally be executed in a few milliseconds, but the communications with micro 2 could exceed 1/2 second. This did not affect the accuracy of the current time variable as the interrupt would halt communications to update that variable. It did, however, cause the counting variables to "miss a beat", since the microprocessor would return to the communications branch after the interrupt instead of repeating the main program loop. The counters could have been adjusted to compensate for this, but the effects were too insignificant to be of concern. The data logging, for example, was supposed to occur every 30 minutes but actually occurred every 30 minutes and 8 seconds. A slight adjustment was made to the counting variables to accommodate the control signal output routine, as this routine was executed more often and could have caused a more significant error. The output pulse sent to the E/P transducer was less than 1/2 second in duration; however, if some of the longer branches of the main loop were executed just prior to the output branch the total execution time for the main loop could exceed 1/2 second. If this happened the interrupt would occur while a control signal was being sent to the transducer and the duration of the control signal would be increased by the amount of time it took to execute the interrupt service routine. Since the signal duration was being used as the controlled variable, this was undesirable. To prevent this an "idle" statement was inserted immediately prior to the output branch so that the full 1/2 second period between interrupts was available for this branch. Since this branch was executed every 30 seconds (60 interrupts) the counting variables were adjusted for the fact that they would "miss" every 60th interrupt.

4.1.2 Main Program As shown in Figure 4.1 the main Emaster program starts by activating the clock interrupt, setting several variables to their initial values, and opening communications with micro 2. It then enters a loop where it sets a "maxoa" variable to a value which will limit the system to running on minimum oa. It also sets counting variables for the data logging and setpoint calculating branches to their initial values, checks to see if the supply fan is running, and repeats the loop if the fan is off. This loop is required because system ACP-7 is normally shut down at night, and when it starts up again in the morning the deck temperatures and flows are changing too rapidly to allow for a meaningful optimization routine. The required mixed air temperature for minimum outside air can be calculated using only the outside and return air temperatures, hence the system is locked in this position until conditions stabilize.

If the fan is turned on the system checks a counting variable to see if 1/2 hour has passed since it last logged data. (Or since the morning start-up, whichever occurred more recently.) If 1/2 hour has passed it transmits operating data to micro 2 so that it can be recorded for later analysis. (The data logged includes the current time, outside air temperature, return air temperature, mixed air temperature, cold deck temperature, hot deck temperature, cold deck flowrate, hot deck flowrate, optimum mixed air temperature, and optimum percent outdoor air.)

Once the data logging step has been completed or bypassed, the counting variable which controls the calculation of the optimum setpoint is checked to see if 15 minutes have passed since the last calculation. If so, a subroutine named "Outair" is called to calculate the new optimum percent outside air and optimum mixed air temperature. After this calculation is completed the "maxoa" variable is reloaded to allow subsequent optimization calculations to call for as much outdoor air as the dampers will physically permit. (As mentioned in Chapter 3 the dampers will only admit up to 85% outside air. If the microprocessor tried to call for, say, 100% outside air an uncontrollable situation would result.)

The final branch in the main program begins by checking to see if 30 seconds have passed since the last control signal was generated. If so, a subroutine named "Control" is called to calculate the required signal. The return air temperature is then compared to the outside air temperature to see whether the control action should be direct or reverse acting, and the sign of the control output is changed if reverse action is required. (Direct action is required when the outside air is cooler than the return air. In this case, a rise in the mixed air temperature would be countered by an increase in the air pressure to the damper motors, thereby bringing in more cool air. If the outside air is warmer than the return air, a reverse acting controller is required so that the outside air dampers will close if the mixed air temperature rises.) An "idle" statement then forces execution to wait until after a clock interrupt occurs, after which the control signal is sent to the E/P transducer.

The main program loop ends with an "idle" statement, execution halts here until a clock interrupt occurs and then loops back to check the fan status and repeat the loop. During morning start up the counting variables are set to values which force the program to calculate a new setpoint (which is limited to minimum outside air) during the first pass through the loop. Fifteen minutes later the setpoint is recalculated, this time allowing for full operation of the dampers. The system is given fifteen minutes to stabilize at this new operating point, then the first data logging occurs. A new setpoint is calculated as soon as the data logging is completed, and normal timing intervals are followed for the rest of the day.

4.1.3 Subroutine Outair A subroutine titled "outair" was used to calculate the optimum percent outside air and the associated mixed air temperature. This subroutine is essentially the same as the optimization routine in program Pdeck (see Figure 2.3), although of course it is written in assembly language rather than in Fortran. A flow chart for this subroutine is given as Figure 4.2. Since the basics of the subroutine were described with program Pdeck, only a few points unique to the microprocessor version will be described here.

The subroutine begins by copying data from the main program into the workspaces which will be used by the subroutine. A workspace is a block of memory which is used as a "scratch pad" by the microprocessor. Data can be transferred to and from the workspace very efficiently and most arithmetic operations require the data be stored in "registers" (16 bits of memory which hold one word of

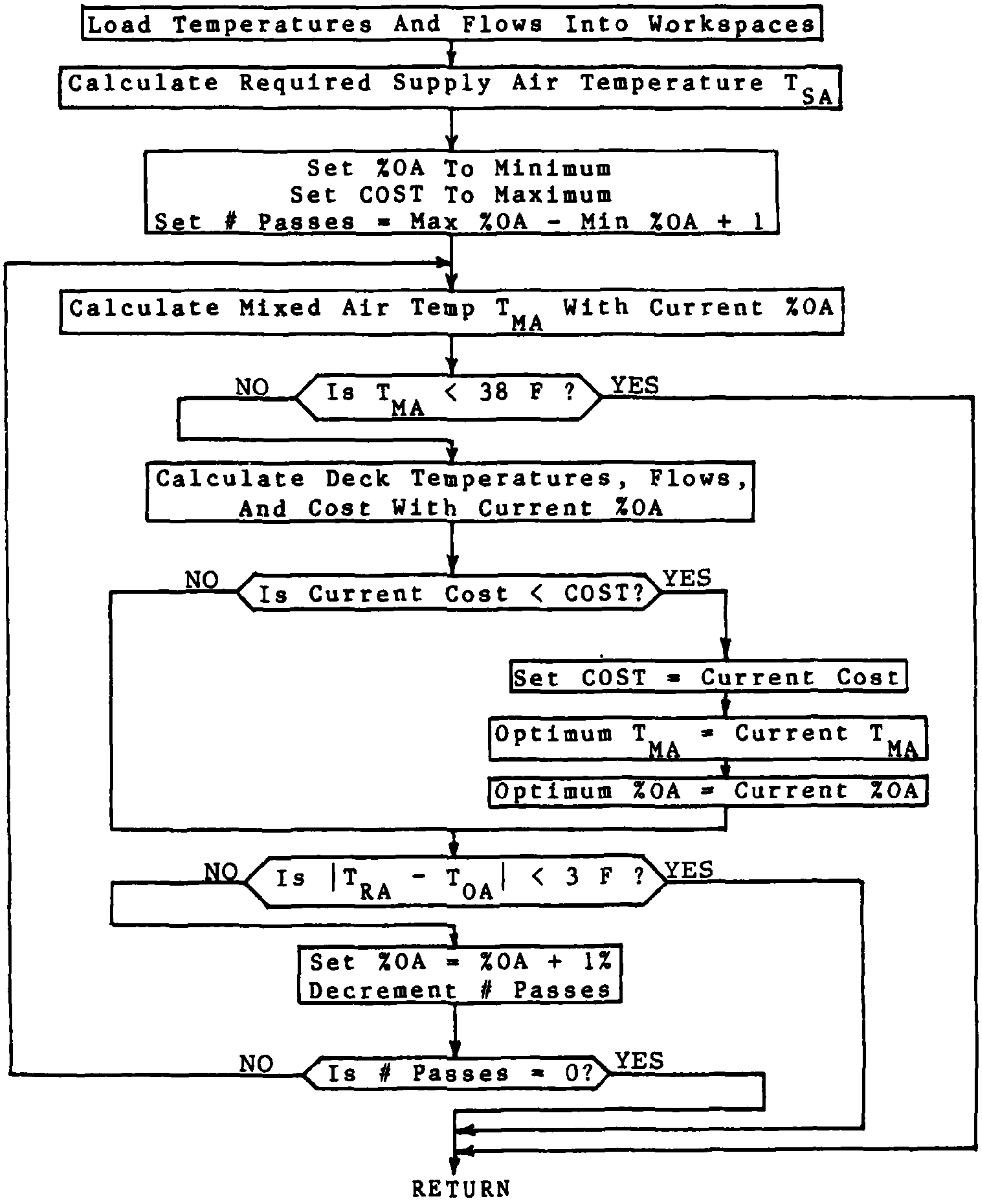


Figure 4.2

Subroutine OUTAIR, Program EMASTER

data) within the workspace. The TI microprocessor uses workspaces which contain 16 registers, so 16 data words can be stored in a workspace. The location of this workspace within the memory is specified by a workspace pointer, operations can be shifted to a different workspace by changing the memory address contained in this pointer. Normally subroutines use workspaces which are different than that used by the main program, but this is not a requirement. When a subroutine is called, the microprocessor stores information it needs to return to the main program in three of the subroutine registers, so 13 registers are available for subroutine calculations. Subroutine Outair required more that 13 registers for efficient operation, so two workspaces were used in this subroutine.

Once the workspaces are loaded the required supply air temperature is calculated from the measured deck temperatures and flows. This is identical to the procedure followed in Pdeck; however, since the microprocessor only works with integers a "round up" check is included after some division operations. When the microprocessor divides one integer by another it stores the quotient in one register and the remainder in another. Subsequent operations generally look only at the quotient, so this has the effect of truncating the quotient. Normally this is not significant, but when data from Pdeck was tested in Outair it was found that the calculated optimum outdoor air could be in error by two or three percent because of successive truncations. For this reason critical division operations in Outair are followed by a routine which adds 1 to the quotient if the remainder is greater than 1/2 the divisor.

After the required supply air temperature has been calculated, data to be used in the optimization loop are set to their initial values. These data include the current percent OA, which is set to the minimum allowable, and the maximum permissible percent OA, which is set to minimum OA during morning start-up and to the maximum possible value at all other times. The difference between the minimum and the maximum is used as a counter to determine how many passes through the loop are required to test all allowable air mixtures. The "cost" variable is set to an unreasonably high figure and the optimization loop will reset this variable each time a lower cost is calculated until it eventually contains the lowest possible cost.

The optimization loop itself is basically the same as that in Pdeck, but two conditions for exiting the loop are slightly different. Program Pdeck operated on the assumption that the return air temperature would remain at 75 F whenever economizer operation was feasible, and used a 72 F outdoor air temperature as a high limit cut-off. Since the actual return air temperature measured by the microprocessor can stray from this 75 F point, subroutine Outair compares the outdoor and return air temperatures and limits operation to minimum outside air if the outdoor air rises to within 3 degrees of the return air temperature. Since the mixed air temperature required to provide this minimum outside air must be calculated before the subroutine is exited, this test is placed at the end of the optimization loop. A form of freeze protection is provided by a command which terminates the optimization loop whenever the calculated mixed air temperature drops to 38 F. Since the optimization loop starts at the minimum percent outside air and gradually increases this percentage, the optimum mixed air temperature will always be greater than this 38 degree cut-off. (In cold weather the mixed air temperature will be warmest when the system is operating at minimum outside air and will drop as the percent outside air is increased.) Theoretically the optimum mixed air temperature would never drop this low unless the cold deck temperature dropped below 38 F, so this cut-off was primarily inserted to prevent coil freeze-up in the event of sensor failure.

4.1.4 Subroutine Control The polarity and duration of the control pulse which was transmitted to the E/P transducer was calculated by a subroutine titled "Control". This subroutine used a standard proportional plus integral plus derivative (PID) control scheme with a few modifications designed to correct problems caused by non-linearities in the dampers. The theory of PID control and its implementation in sampled data systems (i.e. systems where measurements and corrections are made periodically rather than continuously) are well described in many texts. The discussion in this thesis will be very brief and is only intended to help explain the modifications required for this experiment.

The optimal mixed air temperature calculated by subroutine Outair was used as the "setpoint", or ideal condition, and the objective of subroutine Control was to adjust the dampers until the actual mixed air temperature matched this setpoint. Any difference between the ideal and the actual condition is defined as the error, so:

$$\text{ERROR} = \text{SETPOINT} - \text{ACTUAL} = T_{MA_{optimal}} - T_{MA_{actual}} \qquad 4.1$$

Where of course T_{MA} is the mixed air temperature. If the system were operating ideally, the error would be zero, and of course the larger the error is the more the dampers need to be adjusted. An obvious first step in computing the control output, then, is to make the output proportional to the error. With proportional control:

$$P_{out} = K_P\, E \qquad 4.2$$

where

P_{out} = Control Output From Proportional Controller

K_p = Proportional Gain Constant

E = Error (from equation 4.1)

Proportional control (as implemented by pneumatic and electric controllers) has been the mainstay of the HVAC industry for many years. It works very well at bringing the controlled variable close to the desired state, but a small error almost always exists when straight proportional control is used. Until recently this error was not usually considered to be significant, but the rising cost of energy has made it very expensive to operate systems with even a small error. To reduce or eliminate this error the use of integral control has become quite popular. In an integral control scheme the control output is based on the integral of the error over time, so that even a very small error will eventually cause a control output large enough to force corrective action. In this scheme:

$$I_{out} = K_I \int_0^t E\,dt \qquad 4.3$$

where

I_{out} = Control Output From Integral Controller

K_I = Integral Gain Constant

Integral control is relatively slow to react and tends to be unstable so it is almost never used by itself. Instead it is combined with proportional control so that the proportional controller can react quickly to correct large errors and the integral controller can react more slowly to correct small errors.

HVAC systems tend to react rather sluggishly, it generally takes a rather large control "push" to initiate any corrective action but once they begin to react a much smaller control signal will suffice. For this reason derivative control is sometimes added to the PI control scheme just described. Derivative control has no effect on a system that is stationary but will oppose any change in the error with a control output which is proportional to the rate of change. It is described mathematically as:

$$D_{out} = K_D \frac{dE}{dt} \qquad 4.4$$

where

D_{out} = Control Output From Derivative Controller

K_D = Derivative Gain Constant

$\frac{dE}{dt}$ = Derivative of Error with respect Time

If the error is decreasing with time (i.e. the actual value is approaching the setpoint) the derivative $\frac{dE}{dt}$ will be negative and hence D_{out} will be negative. This causes derivative control to oppose any corrective action so it is not used by itself but is used instead to "slow down" corrective motion initiated by the proportional and integral control schemes. This helps keep the system from overshooting its desired position. The output of a PID controller is the sum of the three control equations 4.2 through 4.4, so

$$PID_{out} = P_{out} + I_{out} + D_{out} \qquad 4.5$$

where of course PID_{out} is the output of the PID controller.

The concept of a PID controller is not new, but until recently these controllers have not been widely used in HVAC systems because a mechanical PID controller is much more expensive than a straight proportional controller. The increasing use of computer based controls has made the PID control scheme much more attractive, since the cost difference between a simple control scheme and a complicated one is almost negligible once the computer has been purchased. Since computer control schemes in general (and the microprocessor scheme used in this experiment in particular) do not measure the error and take corrective action continuously but instead perform these functions at discrete time intervals, equations 4.1 through 4.5 are modified slightly for use in a computer. Defining:

$$E_n = SETPOINT_n - ACTUAL_n \qquad 4.1a$$

as the error at time interval number "n", it is easy to define the proportional output at time "n" as:

$$P_n = K_P \, E_n \qquad 4.2a$$

To determine the integral output, it is necessary to use some approximation technique to evaluate the integral of the error over time. The easiest approximation to make is a first order rectangular rule integration where:

$$\int_0^{t_n} E\,dt = \sum_{i=1}^{i=n} E_i \Delta t$$

substituting this into equation 4.3 yields

$$I_n = K_I \sum_{i=1}^{i=n} E_i \Delta t \qquad 4.3a$$

where of course Δt is the time interval between samples. Using a similar first order approximation for the derivative:

$$D_n = K_D \frac{E_n - E_{n-1}}{\Delta t} \qquad 4.4a$$

Substituting these expressions into equation 4.5:

$$PID_n = K_P E_n + K_I \sum_{i=1}^{i=n} E_i \Delta t + K_D \frac{E_n - E_{n-1}}{\Delta t} \qquad 4.5a$$

As long as the time period Δt remains fixed, it can be combined with the integral and derivative gain constants by defining

$$K_I^* = K_I \Delta t$$

and

$$K_D^* = \frac{K_D}{\Delta t}$$

Also, instead of storing all previous error terms in memory the sum of the integral gain constant times each previous error term can easily be stored in a single variable (called "PREINT" in subroutine Control). Thus:

$$PREINT = K_I^* \sum_{i=1}^{i=n-1} E_i$$

Substituting these definitions into equation 4.5a yields:

$$PID_n = K_P E_n + K_I^* E_n + PREINT + K_D^*(E_n - E_{n-1}) \qquad 4.6$$

This was the basic PID algorithm implemented by subroutine Control. Due to non-linearities in system ACP-7 and in the basic control scheme, some of the terms used in this algorithm were set to zero under certain circumstances. For example, when subroutine Outair computed a new optimum mixed air temperature the setpoint used in the control algorithm could undergo a discontinuous jump and previous calculations would become meaningless. For this reason subroutine Control set the previous error (E_{n-1}) and integral sum (PREINT) terms to zero whenever the setpoint changed by 1/2 degree or more. A flow chart of subroutine Control is shown in Figure 4.3. This flowchart is very straightforward and will not be described on a line by line basis. Instead the various non-linearities which caused the PID terms to be altered will be discussed and the sections of the flowchart which implement these changes will be described as appropriate.

The temperature sensors used in this experiment were fairly accurate, nevertheless they were still subject to a certain amount of noise and calibration drift. As

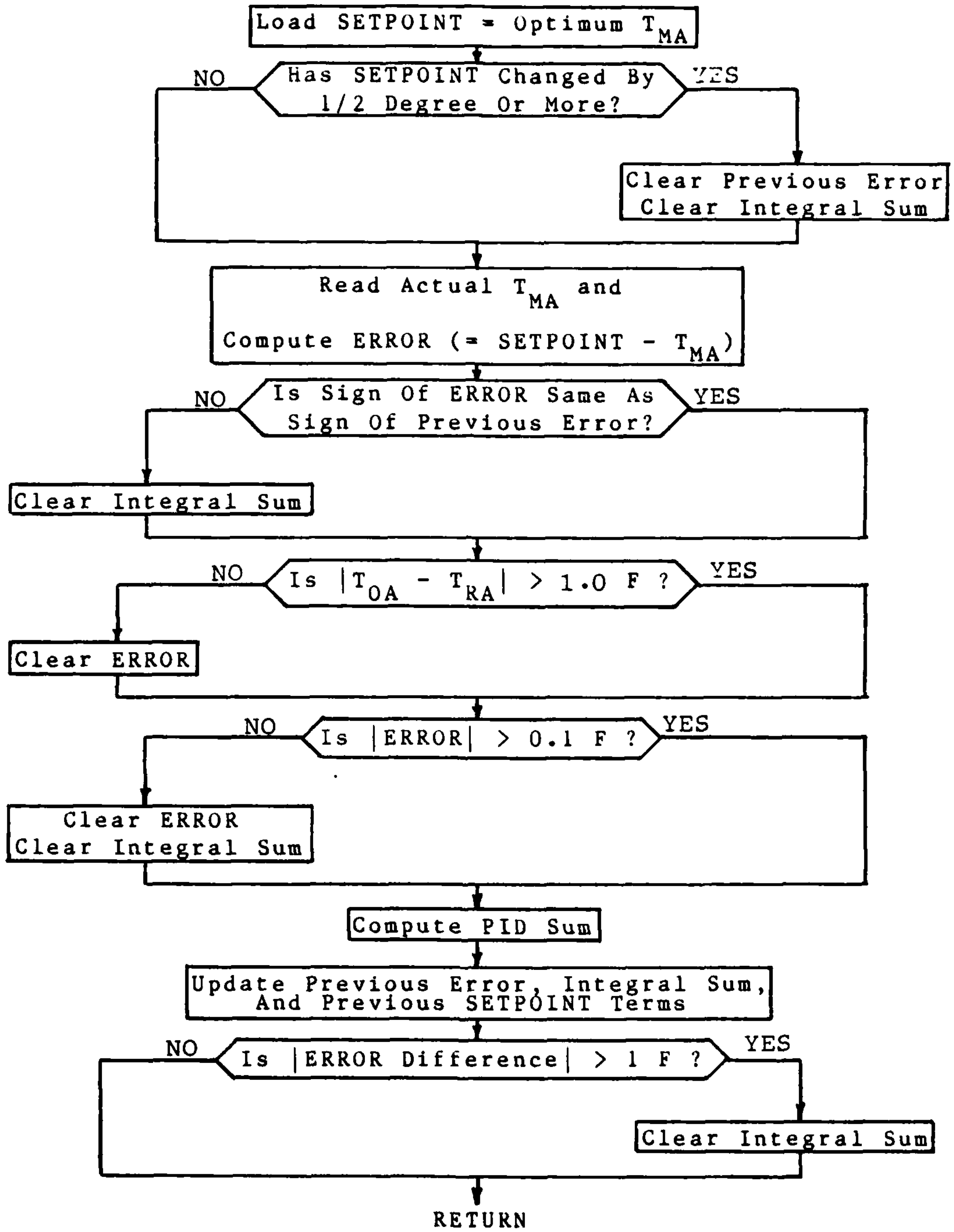


Figure 4.3

Subroutine CONTROL, Program EMASTER

described in the previous chapter the sensors read the temperature to the nearest tenth of a degree. Noise problems occasionally caused the reading to momentarily change by this amount, very rarely did the fluctuation exceed 1/10th of a degree. To prevent this from upsetting the control equilibrium, subroutine Control ignored any error of 1/10th degree or less. This was done by setting the error term E_n and the integral sum PREINT to zero if the absolute value of the error was less than or equal to 0.1 degree.

The slight drift in sensor calibration did not normally cause a control problem; however, when the outdoor temperature and the return air temperature are nearly equal any calibration errors become critical. If the difference between these sensors is, say, 1 degree the mixed air temperature would only change by 1 degree if the dampers moved from fully opened to fully closed. Under these circumstances a 1/2 degree error in the mixed air sensor would cause the percent outdoor air to be in error by 50%. Fortunately, if the outdoor and return air temperatures are that close, an error in the percent outdoor air admitted would not have a drastic effect on the system economy, but still the situation is undesirable. Subroutine Outair called for minimum outdoor air whenever these two temperatures were within 3 degrees of each other, and subroutine Control stopped trying to control the dampers if the temperatures approached within 1 degree. (This was accomplished by setting the error term E_n to zero. This would also insure the integral sum would be set to zero in a later step.) As long as the outdoor air temperature did not change too rapidly, the limit in Outair would first cause the dampers to close to their minimum position and the limit in Control would then shut down the control algorithm and leave them in this position.

A problem which arose much more frequently was that of hysteresis and lag in the dampers. When the PID parameters of a linear system are "tuned" properly the controlled variable will normally overshoot the setpoint slightly if it is reacting to a large initial error or a setpoint change. This is not a problem with a linear system, as the controlled variable will quickly reverse its direction and settle to the setpoint. The time lag in the damper controls; however, caused the measured mixed air temperature to lag behind the actual mixed air temperature by several seconds, so if the dampers were moving rapidly the overshoot could grow quite large before it was detected. The hysteresis in the dampers meant that many control pulses in the reverse direction were required to take up the slack before the dampers themselves began to move. This problem was particularly aggravated if "wind up" occurred, i.e. if the error had persisted a long time before the overshoot occurred and the integral sum had grown so large that it kept "pushing" for more overshoot even though the error was now reversed. To help prevent this a test in subroutine Control set the integral sum to zero whenever the error reversed itself. To help prevent the overshoot from occurring in the first place another test set the integral sum to zero whenever the error changed by more that 1 degree between samples. This had roughly the same effect as using a large derivative gain constant but did not cause the instability problems that a large gain would have. This instablility is caused by the fact that in a sampled data system such as this the derivative term is only computed based on the current and previous error. If the difference is large and the derivative gain is also large, the derivative term can be so large as to cancel out the proportional and integral terms. The control output is therefore zero, no corrective action occurs during the next interval, and the error change is zero. The next derivative term is therefor zero, so the proportional and integral terms will force a large correction. A start/stop motion results, and if the derivative term is very large the system may even reverse its motion every period.

One other problem which should be mentioned was not caused by the nonlinearities in the controlled system but by overflow during addition and multiplication operations. In the previous chapter it was mentioned that this microprocessor worked with sixteen bit numbers and reserved the most significant bit to indicate the sign of the number. (A "1" in this bit indicated a negative number while a "0" in this bit indicated a positive number.) The addition and multiplication routines pre-programmed in the microprocessor did not, however, prevent this bit from being written into and as a result two positive numbers could be added and the sum would be negative. As an example:

```
binary  0111111111111111     (=32,767 in base 10)
      + 0000000000000001
        ----------------
        1000000000000000
```

Of course the answer to 32,767 + 1 should be 32,768, but since the binary number has a "1" in the first column it is interpreted as -32,768. Needless to say, this could cause tremendous control problems. To prevent this subroutines called "Add" and "Mult" were written to perform addition and multiplication with overflow protection. If overflow occurred within these subroutines, the answer returned by them was set to the largest positive or negative (as appropriate) number which could be written with 16 bits. These subroutines were not as efficient as the pre-programmed routines, so they were only used when the nature of the calculation was such that overflow might occur. The results of program Pdeck were very useful in predicting the range of values that the variables in program Emaster would assume. This was used to predict which calculations would need overflow protection.

4.1.5 Other Subroutines In addition to the subroutines already described, program Emaster used subroutines titled "Readt" and "Hexdec" as well as the interrupt service routine. Subroutine "Readt" read the temperature sensors using the conversion equations described in chapter 4. Subroutine "Hexdec" converted the hexidecimal (base 16) numbers used by the microprocessor into their decimal equivalent before sending them to Micro 2 for data logging. This simplified later data analysis, as the high level computer used to analyze the data was programmed to expect decimal inputs. The interrupt service routine merely updated a variable which contained the 24 hr clock time (as in "1530" hrs for 3:30 pm). None of these subroutines contain logic which is essential to an understanding of the control system so they will not be discussed further.

4.2 Program Esub

The software program used in micro 2 was titled "Esub". A flow chart of the main program is shown as Figure 4.4, and a printout of the program is given in the appendix. This is a very simple program which begins by sending an ASCII "0" to micro 1 to indicate that it is "on". Program Esub then waits for a command from micro 1 to instruct it

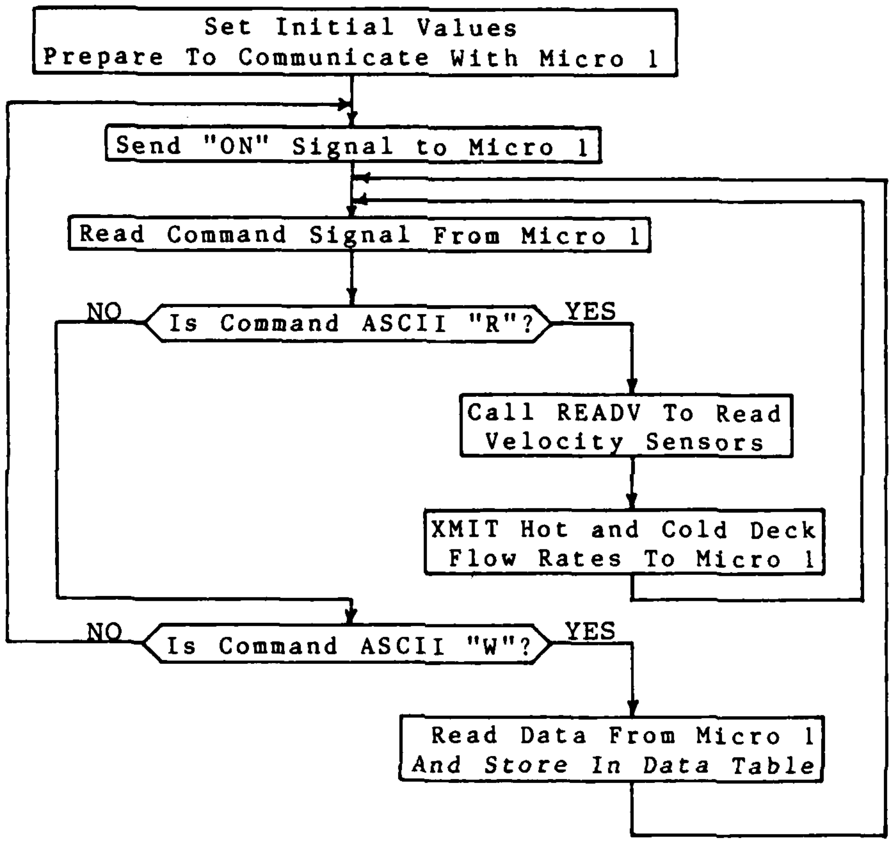


Figure 4.4

Program ESUB (Main Program)

further. If the command is an ASCII "R" (for "read") a subroutine titled "Readv" is called which reads the velocity sensors and converts these velocities to flow readings. This subroutine follows the equations described in chapter 4 and will not be detailed here. Micro 2 transmits these flow readings to micro 1 and then awaits further instructions. If the command is and ASCII "W" (for "write") micro 2 receives 10 data words from micro 1 and writes these data into a block of memory reserved for this purpose. Program Esub then returns to the step where it awaits instructions from micro 1. If programs Emaster and Esub have somehow gotten out of step with each other and the command is neither an "R" nor a "W" program Esub will repeat its "ON" transmission (which alerts micro 1 that an error has occurred) and waits for fresh instructions.

The block of memory which is reserved for data storage is immediately preceded by a short program which instructs the microprocessor to transmit the data through the RS-232-C serial communication port in Fortran 10I6 format. (i.e. 10 data entries per line, 6 digits per data entry. Each data item uses 4 digits, and two spaces are inserted between every pair of data words.) This program is not accessed by the main Esub program and is never executed by micro 2. It is written with the data table to facilitate uploading the data into a high level computer.

Whenever the data was copied onto magnetic tape, this program was copied with it. The tape could then be transported from the mechanical room of the Krannert building (where micro 2 was located) to a room with a computer terminal connected to the Engineering Computer Network (ECN). Here the tape was used to program another Texas Instruments microcomputer, which would then contain both the data and the instructions needed to upload the data into the ECN. This microcomputer was connected to the ECN terminal and was then commanded to execute the upload program.

The data table occupied a block of memory large enough to hold 1,216 bytes of data. Since each data work was 16 bits long (2 bytes) and program Emaster stored 10 words every 30 minutes, this was sufficient to hold 30 hours worth of data. System ACP-7 was normally shut off for 5 to 8 hours each night, so 30 hours of storage was sufficient to allow the times at which the data was transferred to tape to be flexible. The weekend schedule of ACP-7 was such that 30 hours would hold all the data stored from Friday afternoon until Monday morning. Each time the data was transferred to tape and program Esub was restarted it began recording new data at the beginning of data table. After each block of data (10 items) was recorded program ESUB wrote the hexidecimal word "FFFF" in the data space immediately after the last entry. This word was erased when the next data logging occurred, hence it always followed the last data item recorded since the restart and was used by the upload program to indicate the end of the new data. If for any reason the data was not transferred to tape before the 30 hours expired, program Esub would start logging the excess data at the beginning of the data table again and hence the data table always contained the most recent 30 hours worth of data entries.

4.3 Communication Protocol

The "extended operation" (XOP) subroutines written by Texas Instruments were divided into "read" and "write" commands and were further divided into commands which worked with any ASCII character and those which worked with a string of ASCII characters forming a hexidecimal word. XOP #9, for example, was used to read a hexidecimal word. For this reason it expected to receive four ASCII characters and examined them to insure they all were characters 0-9 or A-F (valid hexidecimal digits). Furthermore it expected this word to be followed by a termination character (space, minus, comma, or carriage return) and if anything was received which differed from its expectation it would jump to a "bad data" section of the program. XOP #9 also echoed each character back to the sending computer as it was received. The communication routines in Emaster and Esub were written to take advantage of these features and to repeat an entire communication sequence if anything went astray. All commands and replies, for example, used letters other than A-F so that they could not be confused with hexidecimal characters. Every time one machine issued a command it waited for an appropriate reply from the other machine before continuing with the program. If an appropriate reply was not received it transmitted an error message and then repeated the command. This "transmit-receive-transmit-receive" sequence was used in an effort to prevent any communication problems (noise, fade outs, etc.) from causing both computers to enter a "receive" mode at the same time, a condition in which neither computer would do anything until the other began transmitting. As a final safeguard, the interrupt service routine in micro 1 (a routine which would not be affected by a communication breakdown) transmitted an ASCII "X" every 10 minutes. This would be rejected by micro 2 as an erroneous signal and could cause any on-going communications to be repeated, but it did prevent both computers from spending hours waiting for the other to transmit. This communication protocol was rather time consuming and inefficient, but tests showed that either computer could be stopped and restarted at any point in its program and the communication between the two would eventually synchronize. No communication failures were experienced during the operational use of these computers.

The protocol followed during a typical data transfer is shown in Figure 4.5. Micro 1 initiates the exchanging by sending a "read" command (ASCII "R") to micro 2. Micro 2 is expecting either a "read" or a "write" command, and after it verifies that this is a "read" command it will acknowledge with a "transmitting" reply and then transmit the data. After receiving the "transmitting" reply micro 1 reads the transmitted data and stores it in an appropriate location. In addition, since an xop #9 is used in this step, it echoes the data and the termination character back to micro 2. Micro 2 uses a dummy read xop to clear the echo from its receive register and then transmits a "got it?" inquiry. If micro 1 is satisfied with the data it will reply with a "logged" signal and both programs will continue with their operations. If anything goes wrong and either computer senses an error, that computer will signal the problem with an "error" transmission (generally ASCII "X" or "O") and loop back to the beginning of the communication exchange.

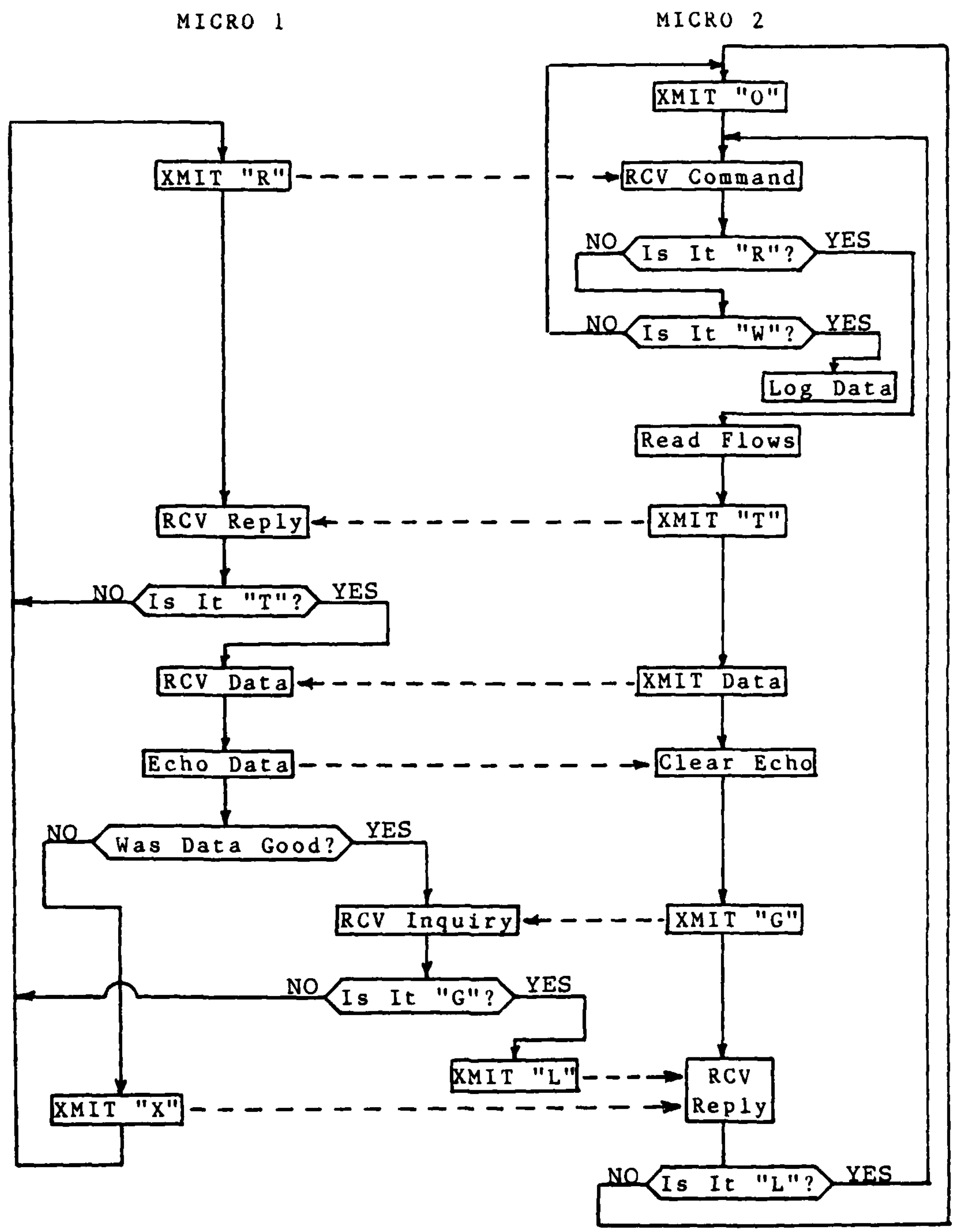


Figure 4.5

Communication Protocol

SYSTEM PERFORMANCE

The experiment described in this thesis was conducted on an operational HVAC system and was therefore subject to many types of random influences and disturbances. These unpredictable events allowed the control logic to be tested under a wide variety of situations, situations which might not have been foreseen in a tightly controlled simulation and which provided invaluable experience in adapting the control logic to a realistic environment. The disadvantage in allowing these random elements to influence the experiment was that they made it much more difficult to interpret the results. The effects of weather and building occupancy will be discussed in the next chapter, this chapter will deal with the effects of the system hardware. Although equipment optimization was not a goal of this research, the performance of this equipment had a definite impact upon the experimental results and thus the system performance will be discussed in terms of how it affected the test results.

5.1 Conventional Controls and Equipment

5.1.1 Hot Deck The performance of the hot deck in system ACP-7 was considerably different than that predicted by program Pdeck. This had a major effect on the economizer cost calculations. In chapter 2 the reset schedule for the hot deck temperature was described. Basically this schedule called for a hot deck temperature of 80 F if the outdoor temperature was below -10 F, a hot deck temperature of 70 F if the outdoor air temperature was above 60 F, and a temperature which varied linearly between these extremes for all other outdoor air temperatures. This curve is plotted in Figure 5.1, along with the hot deck temperatures actually observed during the first 22 days of February. Obviously the hot deck was not behaving as predicted. Physical Plant personnel were aware of this problem and traced it to a low supply air pressure. Since the air being supplied to the pneumatic hot deck controller was at a low pressure, the control output was also at a low pressure and could not completely close the valve which regulated the steam flow into the hot deck. As a result, steam was allowed to flow through the hot deck coils even if the temperature was above setpoint. On 23 February the air supply to this controller was repaired and the hot deck coil was brought under control. The performance of the hot deck during March is shown in Figure 5.2. The hot deck temperature still does not follow the

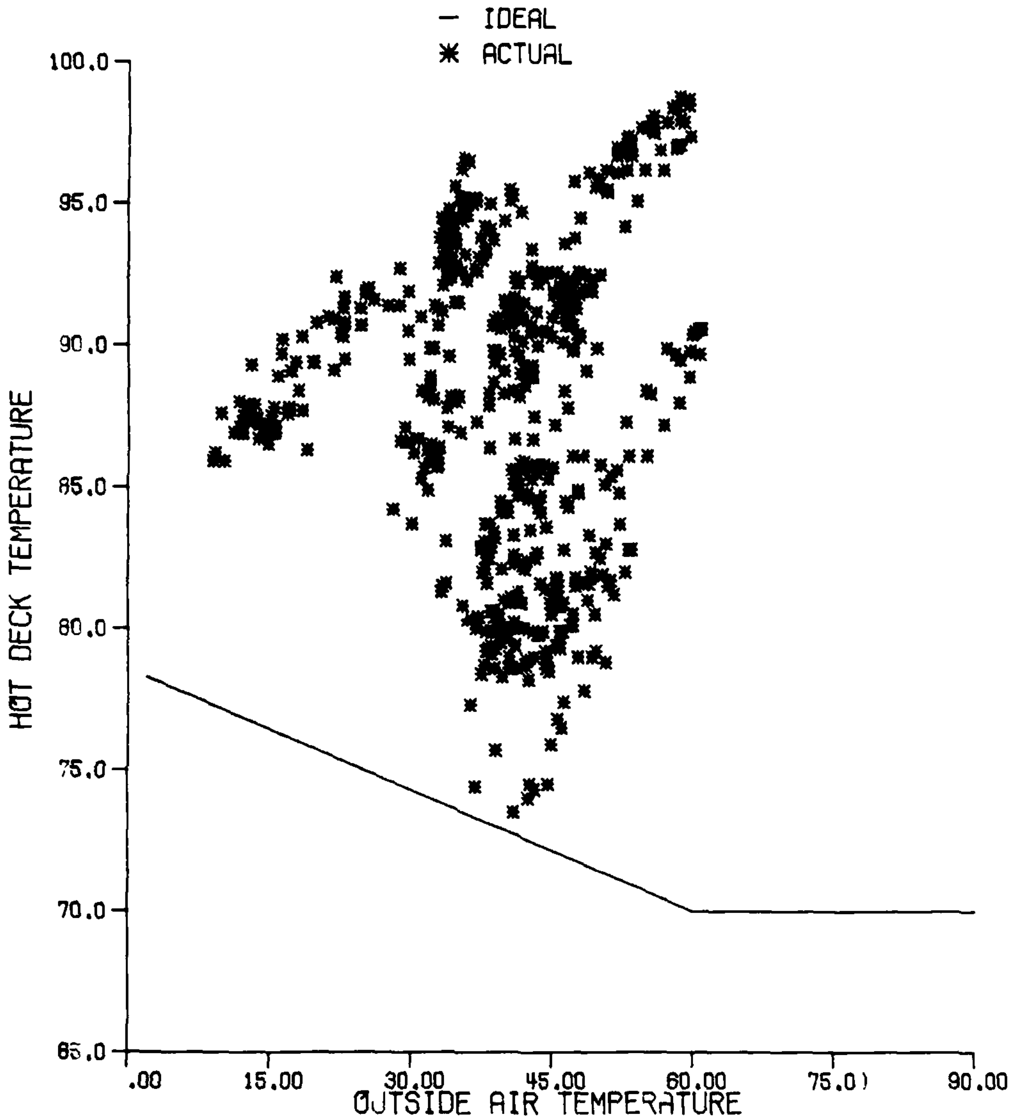


Figure 5.1

Hot Deck Reset

(Before Repair)

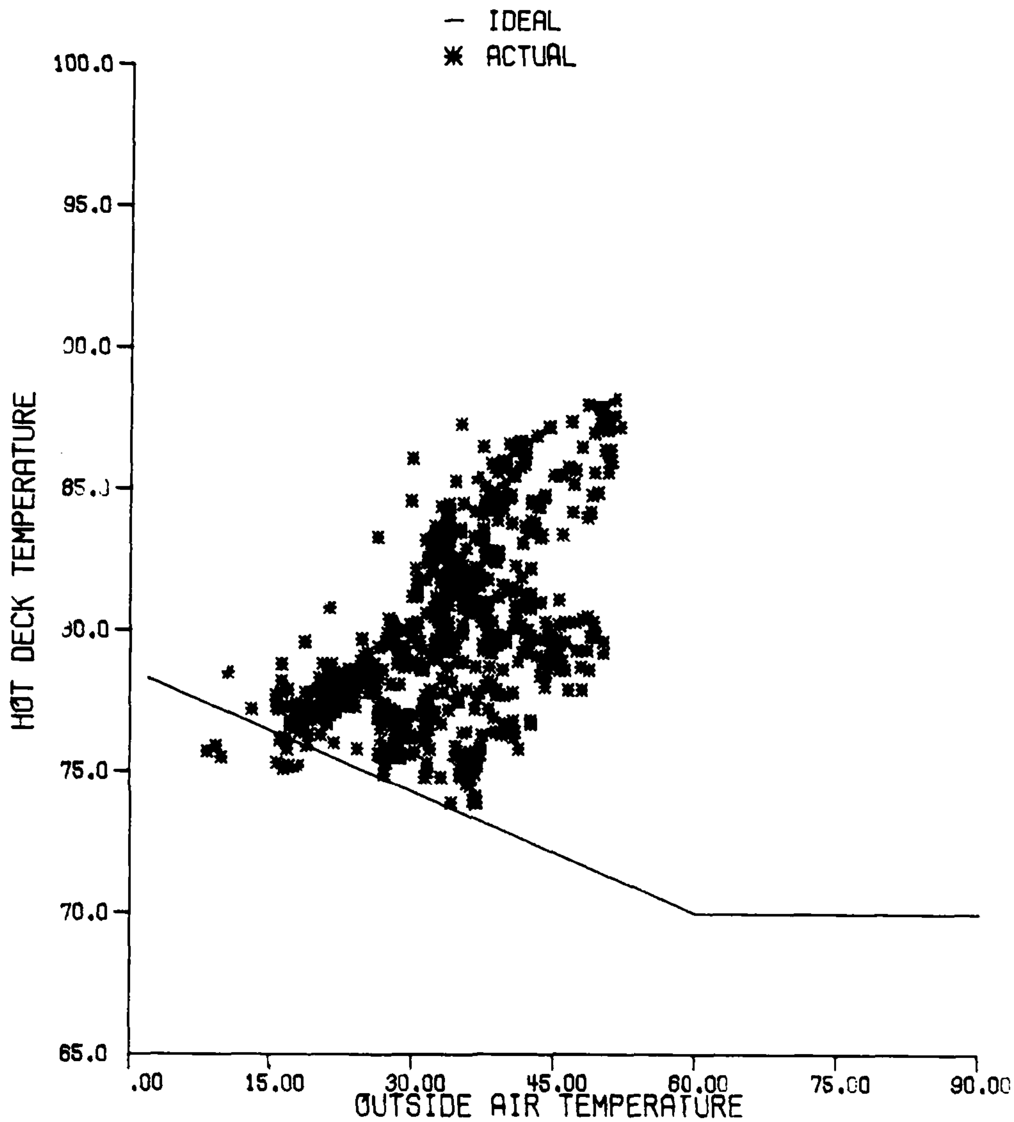


Figure 5.1

Hot Deck Reset

(After Repair)

ideal curve very closely, but its performance is markedly better than when the coil was allowed to run "wild". Since the hot deck temperature was generally greater than the ideal temperature (as used in Pdeck), a smaller airflow was needed to meet the heating load. System ACP-7 basically operates at a constant airflow, so the low flowrate through the hot deck resulted in a greater flow through the cold deck. The elevated hot deck temperature therefore increased both the heating and cooling costs, but since cold deck cost/Btu was greater than the hot deck cost/Btu the cold deck costs increased more rapidly. Thus the net result of operating the hot deck at a higher than ideal temperature was to make economizer operation advantageous at lower outdoor temperatures and/or lower internal heating loads than was predicted by Pdeck.

One of the reasons why the hot deck was allowed to operate at a high temperature was to offset the stratification problem. As mentioned in chapter 2, the air flowing off one end of the hot deck was generally about 15 F warmer than that flowing off the other end. The ductwork split into two main supply ducts immediately downstream of this coil, so very little mixing occurred. Because of this the hot air supplied to some rooms was 15 degrees warmer than that supplied to the other rooms. The temperatures shown in Figures 5.1 and 5.2 are based on an averaging sensor strung across the entire hot deck and were calibrated using the average of four readings (two in each supply duct). In general this average was very near the midpoint between the highest and lowest readings, so a hot deck temperature of 80 F on Figure 5.1 or 5.2 indicates some rooms received air at about 87 F and some received air at 73 F. If the reset schedule had been allowed to drop the average hot deck temperature down to the ideal limit of 70 F, some rooms would have received air at 63 F. This is much too cold to provide effective heating, so the hot deck was maintained at a higher than ideal temperature. Physical plant personnel were aware of the stratification problem, but the fault appeared to lie in the heating coil itself and replacement costs were prohibitive so the problem was not fixed. This problem was unique to the hot deck, neither the mixed air section nor the cold deck ever showed more than one or two degrees of stratification.

The desired room air temperature for the area serviced by ACP-7 was 75 F, hence the supply air had to be warmer than this to provide heat. Due to the stratification problem, this required the average hot deck temperature be greater than 82 F. Figure 5.2 shows that the hot deck temperature often dropped below 82 F, so some rooms received supply air at less than the 75 F minimum. The reason these rooms did not drop to unacceptably low temperatures was because they were also being heated by a separate hot water system. This system supplied 100 F water to radiators located around the perimeter of the building and effectively countered most heat loss to the outside. (Only the rooms on an outside wall contained these radiators, the rooms not on the perimeter did not need them because they were surrounded by warm rooms and had very little heat loss.) This hot water system was in operation 24 hours per day during the winter months and thus the rooms did not get cold even when system ACP-7 shut down at night. During the night of 6/7 February, for example, the outdoor air temperature dropped to about 9 F, yet when system ACP-7 started up in the morning the return air was 74.8 F. When the data was first logged that morning the cold deck flowrate was almost twice that of the hot deck, indicating the two heating systems were suppling much more heat than was actually required. Thus the net effect of this perimeter heating system was to greatly increase the fixed internal cooling load and thereby extend the range during which economizer operation was feasible.

5.1.2 Cold Deck The fact that the hot deck temperature did not follow the ideal reset schedule used in program Pdeck did not require any changes to the decision making algorithm. The optimization routine assumed that the measured hot deck temperature was in effect the hot deck setpoint and would remain constant regardless of the percent outside air admitted. This assumption was not entirely correct, the error tolerated by the pneumatic hot deck controller would have let the hot deck temperature vary somewhat as the mixed air temperature varied, but since the hot deck temperature was always considerably warmer than the mixed air temperature this did not greatly influence the results. The error tolerated by the cold deck controller could not, unfortunately, be handled as easily. Ideally when economizer operation was feasible the economizer would provide mixed air at the cold deck setpoint so that the cooling coil could be shut off completely. In reality, however, the error tolerated by the controller (generally referred to as "offset" in control literature) would cause the cold deck temperature to drop below its setpoint if the mixed air temperature was lowered to this setpoint. Thus if the measured cold deck temperature had been used as the cold deck setpoint the economizer would have controlled the mixed air temperature to this value during its first cycle. During the next 15 minute cycle the cold deck temperature would have dropped below this setpoint, so the economizer would drop the mixed air to the new cold deck temperature. This "ratchet" effect would have eventually caused the economizer to maintain the cold deck at a much lower temperature than was required and would have increased the corresponding hot deck cost. This would have been undesirable, so the measured cold deck temperature was not used as the cold deck setpoint. Instead a fixed cold deck setpoint of 63 F. was used for economizer calculations. This temperature was based upon measurements taken while the system was operating on minimum outside air and was set slightly above the average cold deck temperature so that the economizer would not override the cold deck controller. This prevented the economizer from raising the total system cost by forcing the cold deck to operate below its setpoint, but it also prevented the economizer from reducing the cold deck cost to zero. Thus the net effect of using a fixed cold deck setpoint was to produce a "conservative" controller, one which made few mistakes but which also could not reduce the cold deck cost quite as much as the ideal controller in Pdeck. Note that this problem was caused by the marriage of the microprocessor based economizer with the pneumatic deck controllers. If the microprocessor had controlled the decks as well as the dampers (as would be the case with a commercial DDC unit), the offset error would have been largely eliminated and the microprocessor would not have had to "guess" what the deck setpoints were.

5.1.3 Dampers Most of the dampers used in the mixed air section of ACP-7 were of the parallel blade type, only the return air dampers were opposed blade. The outside air dampers were "low leakage" dampers, that is they were fitted with weatherstripping which allowed them to seal more effectively when completely closed. Parallel blade dampers are not normally recommended for the type of variable control required in this research, the flow through them does not vary linearly with the actuator position and this can lead to control problems. This did not seem to cause any major problems in this experiment, as the microprocessor almost always kept the actual percent outside air within one percent of the optimum value. During morning start-up, however, the error was sometimes as large as five percent at the first data logging (15 minutes after the setpoint was calculated). This error could be either positive or negative, indicating the dampers had either overshot their setpoint or had not yet opened far enough. Part of this error could have been caused by the inherent non-linearity of the parallel blade dampers, but a more probable cause is the hysteresis and lag time in the damper actuating system. Figure 5.3 shows the time response of these dampers to both *an increase* and a decrease in the actuating pressure. The manual damper positioning control was used to change this pressure, the procedure was as follows:

Prior to the start of the test the control signal to the dampers was dropped to zero to allow them to close completely. The pressure was then increased to 6 psi, a pressure which roughly corresponds to minimum percent

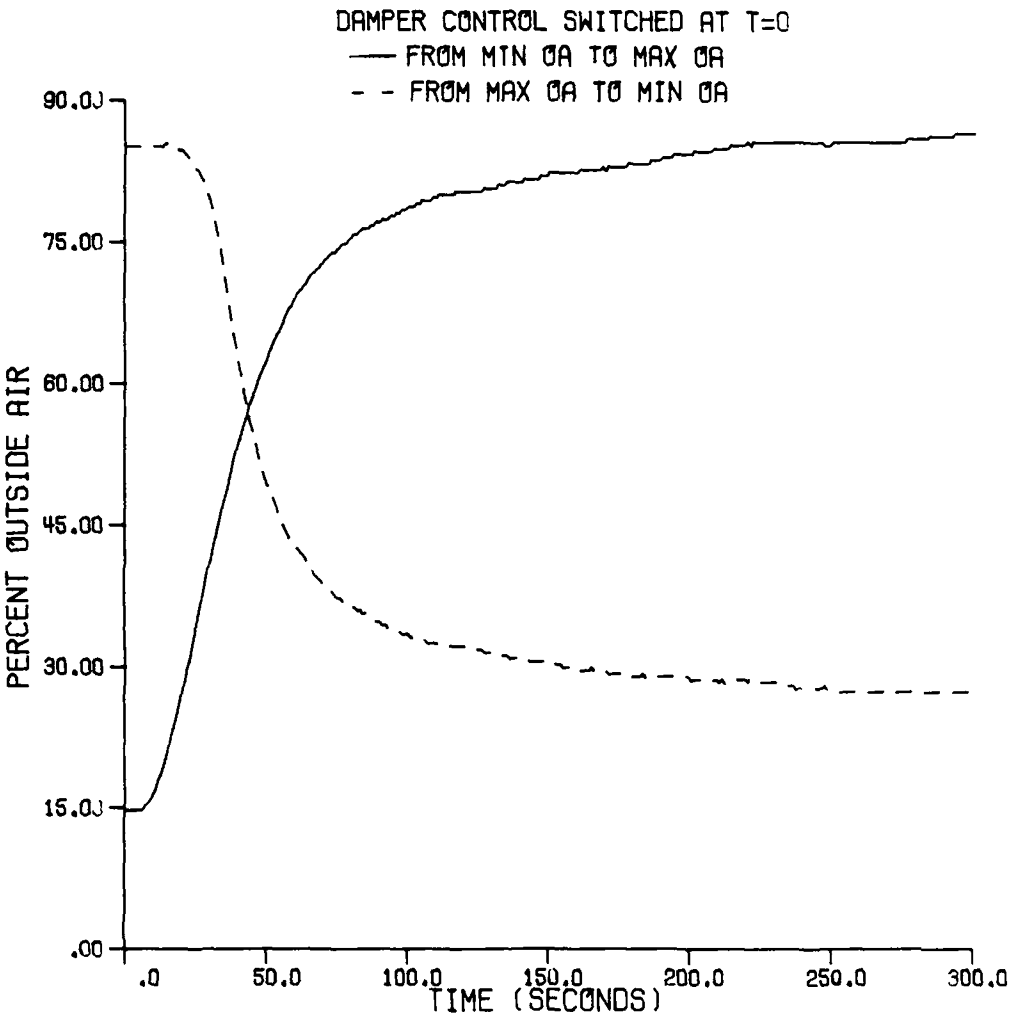


Figure 5.3

Time Response Of Dampers

outside air. The mixed air temperature was allowed to stabilize at this point, then the manual control was rapidly changed to supply an 18 psi control signal at the same time as a microprocessor program which logged the mixed air temperature every second was activated. This program was allowed to run until the dampers were completely opened and the mixed air temperature again stabilized. At this point the manual control was turned back to a position which sent a 6 psi control signal to the dampers and the microprocessor program was again activated. At the beginning of each test the outside and return air temperatures were recorded so that the mixed air temperatures could be converted into percent outside air readings.

The data plotted on Figure 5.3 shows a very pronounced hysteresis problem. When the control pressure was raised to 6 psi from 0 psi it caused the dampers to open to a position which admitted about 15 percent outside air, but when it was dropped to 6 psi from 18 psi the dampers only closed to a position which admitted 27 percent outside air. The lag time is also evident on this graph, when the pressure was increased it took approximately 7 seconds before the effect was noticed and when the pressure was decreased it took about 19 seconds to sense the effects. The longer lag time for the drop in pressure can be explained by the fact that 18 psi is more pressure than is required to open the dampers completely, and the excess pressure had to be bled off before the dampers could begin to close. In both cases it took nearly five minutes before the mixed air temperature stabilized. Note that since Figure 5.3 is based on measurements made by the microprocessor, the temperature sensor lag time and the transport lag (the time it takes the air to flow from the dampers to the sensor) are included with the damper lag time. Both tests were run within a few minutes of each other, so sensor drift and calibration errors did not affect the hysteresis and lag time measurements. The actual hysteresis and lag time the control algorithm had to contend with were worse than that shown in Figure 5.3, as this plot does not include the response of the E/P transducer. This transducer exhibited considerable hysteresis, which made the control response even more sluggish.

5.1.4 <u>Total Airflow Through System</u> The optimization routine used in the program Emaster is based on the assumption that the total flow through system ACP-7 will remain constant regardless of changes made to the percent outdoor air admitted. This airflow will not, in fact, remain absolutely constant but will instead vary as the flow through the individual room supply ducts varies. If, for example, all rooms were calling for full heat, the entire system airflow would be channeled into the heating ducts.

If all rooms were instead calling for a 50/50 mixture of hot and cold air, the system airflow would be divided between the hot and cold ducts. With roughly twice as much ductwork available to carry the flow, the flow resistance would be less in the second example than in the first. The supply fan turns at a constant speed, hence the lower resistance would allow it to move a greater amount of air. The total system airflow will therefor vary somewhat as the individual room loads vary. The microprocessor control scheme will vary the percent outside air admitted to control the mixed air temperature, and this may in turn affect the deck temperatures. (Especially since pneumatic deck controllers are being used.) Since changing the deck temperatures will cause the room demand to vary, the total airflow may change as the microprocessor varies the mixed air temperature.

In order to estimate how significant this variation was, the hot and cold deck flows recorded by the microprocessor were added together to form the system total flow and this is plotted in Figure 5.4. This graph shows the system airflow for the entire month of March. The "time" axis is not to scale, the individual readings are plotted at equal intervals regardless of whether they occurred 30 minutes apart or were separated by a weekend shutdown. The graph shows that the flow through the system is essentially constant, varying by about $\pm$ 10%. This fluctuation

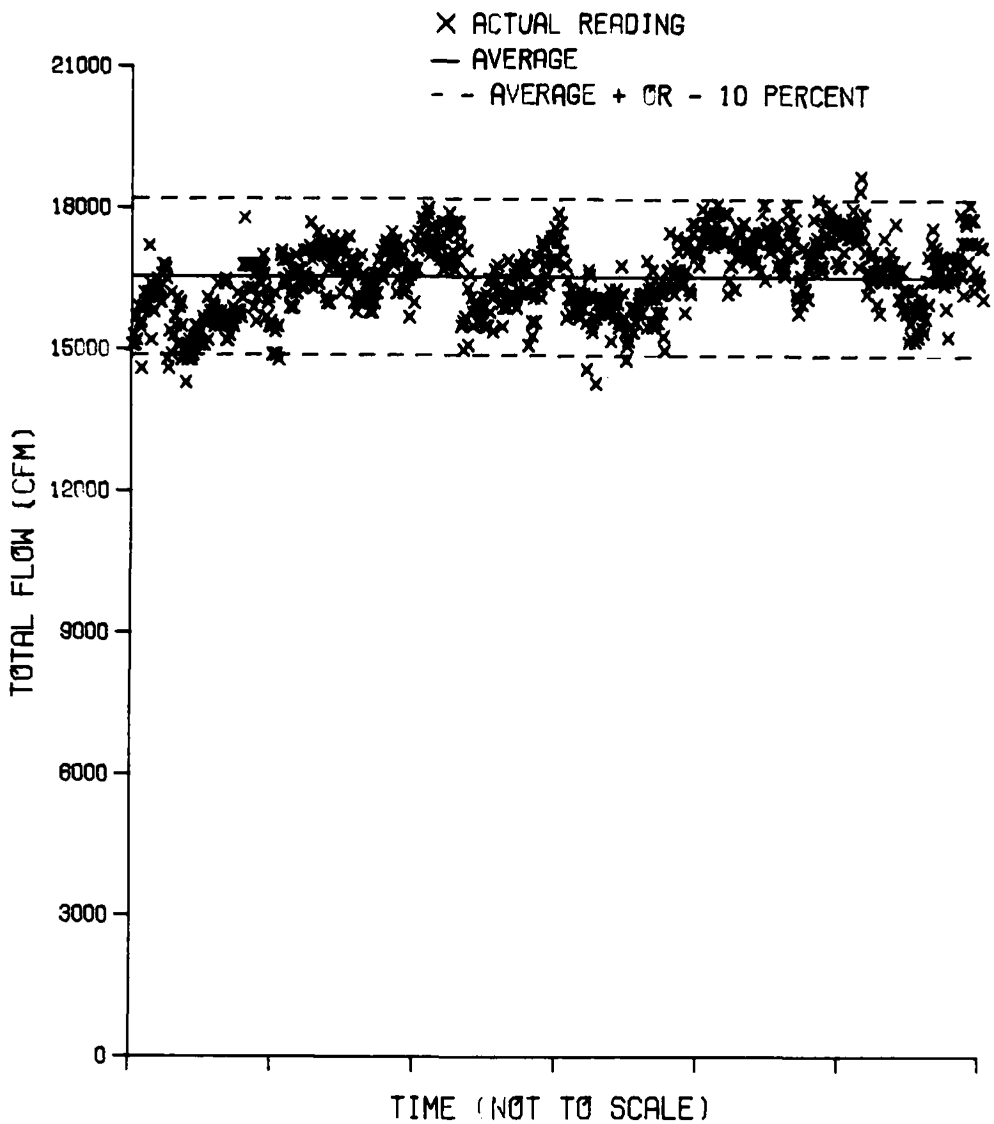


Figure 5.4

Total Flow Through ACP-7

is consistent with the error expected from the individual velocity sensors and does not significantly affect the optimization calculations. At least some of this fluctuation can be attributed to the fact that the flow in the ducts is turbulent and causes the sensor readings to fluctuate around the average flow, hence the actual variation in the average system airflow is probably not as great as Figure 5.4 would indicate.

5.2 Sensors

5.2.1 Temperature Sensors As described in chapter 3, point sensors were used in the outside and return air ducts while averaging sensors were used to measure the mixed air, hot deck, and cold deck temperatures. The calibration of these sensors was checked using a laboratory thermometer on a weekly basis, with a single reading being compared to the point sensors and the average of several readings being compared to the averaging sensors. The outside air sensor initially performed very erratically, the weekly error readings for this sensor are shown in Figure 5.5. Although it appears to be random in Figure 5.5, a pattern to this error was noted during the first several weeks of operation. Whenever the outside air temperature dropped below that at which the sensor had been calibrated, the outside air sensor would indicate a temperature which was too high (causing a positive error).

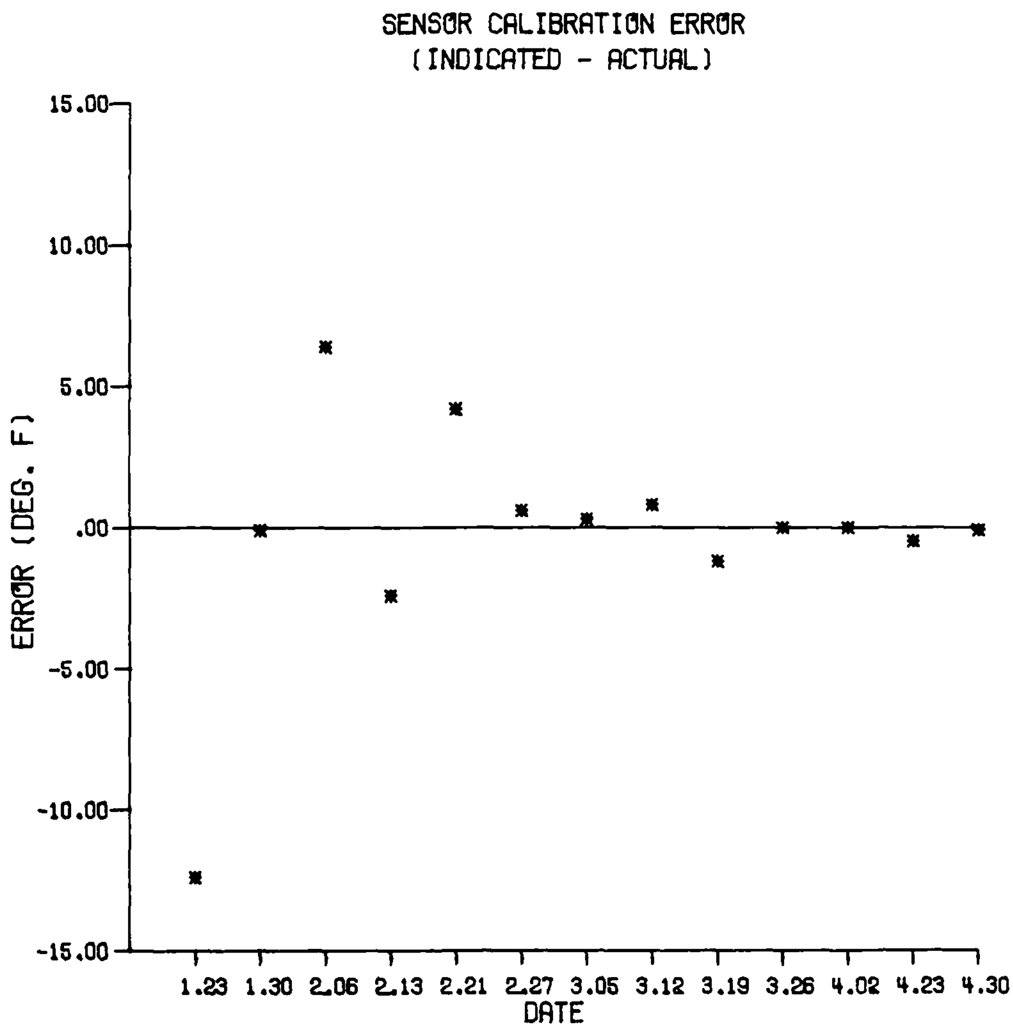


Figure 5.5

Outdoor Air Sensor Error

If the calibration was changed to compensate for this error and the outside air later warmed to a temperature above this new calibration point, the error would be negative. It seemed probable that this error was caused by conduction along the support bracket. As shown in the photograph in chapter 3 (Figure 3.5), the point sensor only extended into the ductwork approximately 5 inches. (The bracket was about 6 inches long, but the duct was sheathed in 1 inch insulation.) This was long enough to clear the relatively stagnant airflow in the boundary region, but it was short enough so that the steel support bracket could conduct an appreciable amount of heat to the sensor. The outside air sensor was particularly sensitive to this conduction, as one end of the bracket was located in a mechanical room at roughly 75 F. while the other end was supporting the sensor in air that occasionally dropped to sub-zero temperatures. The degree to which this conduction could affect the reading would depend not only on the temperature difference but also on the airflow past the support bracket, since this affected the rate at which the conducted heat was dissipated. A quick test of this theory was performed by varying the percent outside air from roughly 20 percent to around 40 percent (thereby increasing the airflow) on a relatively chilly day. This caused the indicated outside air temperature to drop 3 degrees, indicating heat conduction was affecting the sensor readings. The short bracket was therefore replaced by a steel tube approximately two feet long. The sensor element was suspended from the end of this tube by a short length of wire which prevented it from touching the tube. Subsequent tests showed changing the percent outside air admitted had no appreciable affect on the sensor reading. This change was made on 23 Feb, and Figure 5.5 shows that the accuracy improved considerably after that date. (The dates given along the x-axis are given as month.day, hence 2.21 is February 21st.) After this modification had been completed, it was learned that MCC Powers had begun installing similar elongated point sensors as part of a computerized energy management system they were installing at Purdue. Apparently the conduction problem was not unique to this experiment.

The performance of the averaging sensors was, in general, superior to that of the point sensors and the performance of the mixed air sensor was particularly good. This was fortunate, as the accuracy of this sensor was especially critical. Kao and Pierce (23) have shown that a 5 degree error in this sensor can increase the cooling load by 60%. Their research was done on a terminal reheat system and not on a dual duct system, but the emphasis they placed on accurate sensors applies equally to both types of systems. A plot of the mixed air sensor performance is given as Figure 5.6.

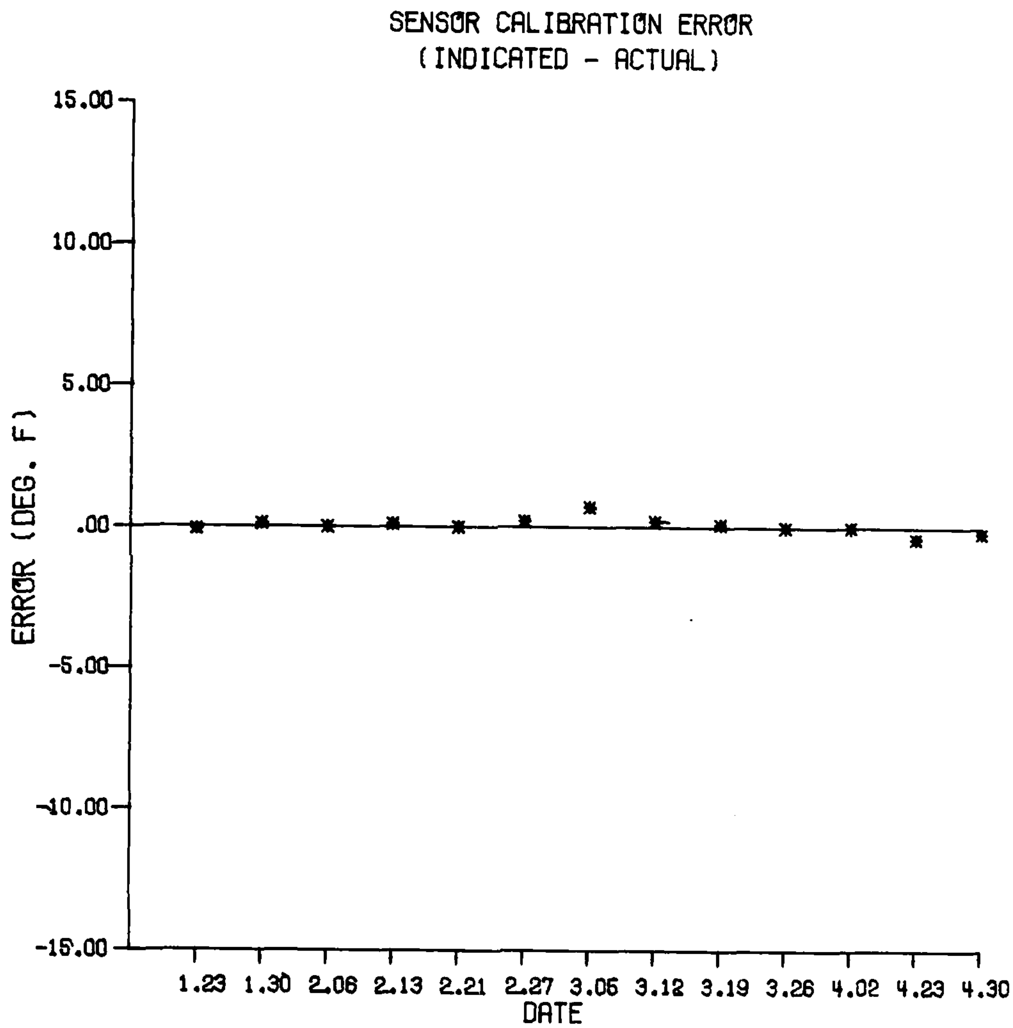


Figure 5.6

Mixed Air Sensor Error

As a final note on the temperature sensors, the calibration procedure used in this experiment was in itself subject to a certain amount of error which could have affected the results shown in Figures 5.5 and 5.6. The laboratory thermometer used to calibrate these sensors was a mercury bulb glass thermometer marked in 0.2 degree increments. The readings were interpolated to the nearest 0.1 degree, a procedure which left some room for error. The airstreams which were being measured were not entirely homogeneous, even the outdoor air and return air ducts showed a variation of several tenths of a degree if measured at different points. An attempt was made to check the air temperature as close to the sensor as possible, but the construction of the ducts did not always allow this. More importantly, the temperatures were varying with time and a certain degree of error was introduced by the fact that the actual temperature could change between the time it was checked with the thermometer and the time at which the sensor reading was checked and adjusted. This could be especially significant in the hot and cold decks, since their calibration required four separate temperature readings, each of which took several minutes, and the deck controllers could change the deck temperatures during this period. Even the outdoor air temperature was not static, changing wind or sunlight conditions could cause the temperature to change several tenths of a degree during the calibration procedure. For these reasons the sensor calibration was usually not changed unless the error exceeded 0.2 degrees, and the calibration plots for the temperature sensors reflect errors in the calibration procedure as well as sensor errors.

5.2.2 Velocity Sensors The performance of the velocity sensors is of particular interest, not because their accuracy was extremely critical but rather because they are unique to this experiment. No type of velocity sensor is widely used in HVAC control applications. Some of the newer variable air volume (VAV) systems require flow measurements, but these systems generally either use pitot tubes or static pressure measurements as a means of sensing airflow. The heated thermistor sensors were selected for this experiment because they were much less expensive than the commercially available sensors, they showed promise of being more accurate at low flow rates, and little was known of their performance in a HVAC application. Spare sensors were initially prepared because it was feared that the fine wires supporting the velocity sensing thermistor (see Figure 3.7) would be broken by airborne dirt particles. There was also concern that this thermistor would become covered with dust and therefore be insulated from the airstream. Neither of these problems disabled a sensor during this experiment, the four sensors initially installed in January were still functioning when the experiment was terminated in May. A plot of the calibration error of one sensor is given in Figure 5.7. This graph shows the performance of a sensor installed in one of the cold air ducts. Its performance is typical of the other velocity sensors; however, this graph is of particular interest because the calibration of this sensor was never changed during the experiment. By leaving the calibration unchanged any slow drift in performance should be apparent in the weekly error readings. There does not appear to be a great deal of drift in these readings; however, the errors do grow increasingly negative as the weeks go by. The magnitude of these errors is not sufficient to be of concern, as it remained below 5% of the total flow. (The flow in this duct was typically around 1000 ft/min.) A negative error indicates the actual airflow is greater than the sensed airflow, so the slight drift visible in Figure 5.7 could have been caused by a dust build up on the thermistor. This sensor was removed from the duct for examination on 26 March. The sensor did not appear to be dirty, and a spray type contact cleaner was used to remove any slight contamination which might have been present. The slight improvement in performance seen on this date in Figure 5.7 may be the result of this cleaning or may be coincidence. A long term experiment would be required to determine if this drift would continue until it became significant.

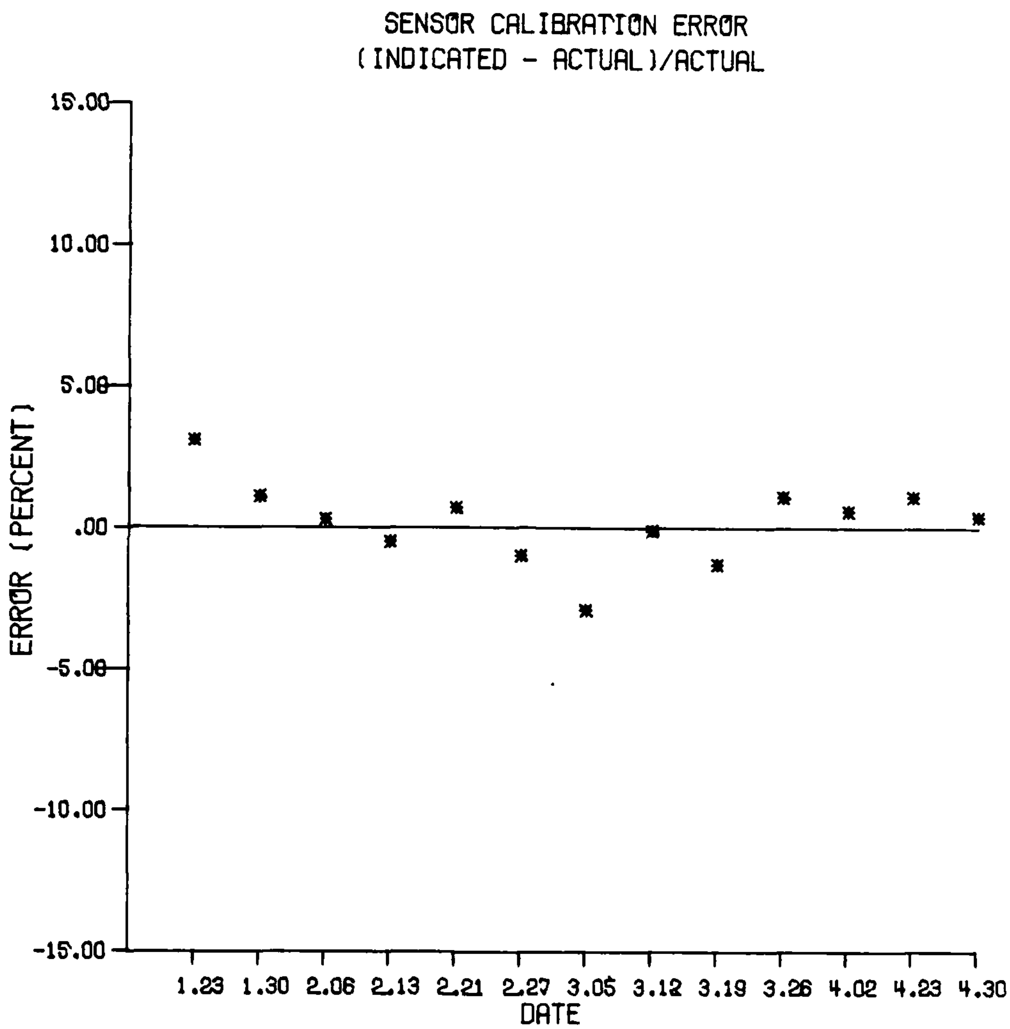


Figure 5.7

Velocity Sensor Error

As was the case with the temperature sensors, errors in the calibration procedure itself could have influenced the results shown in Figure 5.7. Although a fairly well developed turbulent flow profile was present in the duct, the velocity did vary with time and with position so it was impossible to determine exactly what the velocity at the sensor was at the time when it was being calibrated. The hand held velocity meter used to check the calibration could only be read to within ± 20 ft/min, and the microprocessor averaging routine introduced a similar round-off so the errors indicated in Figure 5.7 could have been influenced by the calibration procedure itself as well as by the sensor.

5.3 Control Parameters

The accuracy and responsiveness of the experimental control system was determined by several variables contained within the software as well as by the hardware limitations already described. The variables with the most significant effect were those which determined the timing of control actions and the gains used in the PID algorithm. The effects these variables had on the system were very much interrelated, the optimum value of any one variable depended on what values were being used for the other control parameters. The optimization of the entire set of variables could be the subject of an entire research effort in itself, in this experiment the parameters were only adjusted until the system performed in a reasonably acceptable manner. This section will describe how these variables were adjusted and offer some subjective comments on how they could be optimized.

5.3.1 Timing As described in chapter 4, program Emaster transmitted a corrective signal to the E/P transducer every 30 seconds and calculated a new setpoint every 15 minutes. The fifteen minute time period was recommended by Robert Coughlin (24) in his case study of a direct digital control system. This interval was adopted at the beginning of the experiment and was never changed. It appeared to be a good choice, as even when a major set-point adjustment was made (as in the morning start-up) 15 minutes was generally long enough to allow conditions to stabilize before new computations were made. The dampers usually took at least 10 minutes to adjust to a major set-point change, so a shorter interval might not have allowed the system to operate at any one setpoint long enough to find an optimum operating point. The thirty second inter-val was adopted after shorter periods were tried and found to cause severe overshoot problems. Initially a 5 second interval between control pulses was tried, but even if very small gains were used in the PID algorithm the dampers would overshoot their setpoint by a considerable amount. (Figure 5.3 indicates that a 5 minute interval would be required to prevent overshoot during a major correction, but since the system is not linear this may not hold true for a smaller correction.) The use of small gains to reduce the overshoot meant that a very long period of time was required to overcome the hysteresis before the overshoot could be corrected. Ten and twenty second intervals were tried and found to cause similar problems. The thirty second interval seemed to provide a good compromise between the fast initial response provided by shorter intervals and the good overshoot control provided by longer intervals. This value also falls within the ranges recommended by Jones (25) and Sams and others (26). Again it should be stressed that the PID gains were not optimized for each different interval, they were merely adjusted until a reasonably stable control was achieved.

5.3.2 PID Gain Constants As described in chapter 4, the proportional, integral, and derivative gains used in this control could be adjusted independently. The procedure used to find initial values for these gains is based on the Ziegler/Nichols method as described by Johnson Controls (27) and is as follows:

Initially the integral and derivative constants were set to zero (providing straight proportional control) and the proportional gain was slowly increased until the system became unstable. This was a trial and error procedure, after each change to the gain constant the manual positioning control was used to close the dampers, then the microprocessor control was activated and the E/P transducer output pressure was monitored to determine overshoot. The microprocessor was set to provide minimum outside air and the pressure which corresponded to this position was known from previous experiments. Since the slow response of the system meant that it could take many minutes (or hours) for oscillations to die out, an initial overshoot of 100% or more was assumed to indicate instability. Once the proportional gain which caused instability was found, the gain variable was set to 1/2 this value and the integral gain was slowly increased until instability again resulted. The integral gain variable was then set to 1/3 the value which caused instability and the derivative gain was tested. The derivative gain variable was similarly set to 1/3 the value which caused instability.

Once initial values for the PID gains had been determined by this procedure, small adjustments were made by timing the response of the system to a 10% change in setpoint. The proportional gain was increased if the initial response seemed slow and decreased if the overshoot was excessive, the integral gain was increased if the recovery from a small overshoot seemed slow and decreased if it again overshot the setpoint, and the derivative gain was increased if the initial overshoot was large and decreased if a start/stop motion was detected. The time it took for the system to settle to the new setpoint with each set of gains was recorded and the gains which produced the shortest settling time were adopted as being the optimum. Slight adjustments were made to these gains during the first month of the experiment, as it became apparent that they were too large and caused overshoot if setpoint changes larger than 10% were encountered. The values which were finally adopted as yielding the best performance were (in hexidecimal numbers):

Proportional Gain	=	200
Integral Gain	=	25
Derivative Gain	=	300

The actual values of these numbers are of little importance, as they are probably not the optimum values for any other system and may not be optimum even for this system. What is important is their relative magnitudes. For this system the integral gain had to be much smaller than the proportional gain or excessive "wind-up" resulted. The excessive hysteresis in the system meant that overshoot could be extremely troublesome, so a very large derivative gain was used to slow down the control once the dampers started moving. For the same reason, the proportional and integral gains were smaller than those obtained during the initial optimization test. It should be noted that the system was not especially sensitive to these gains. Changing the integral gain from 25 to 50, for example, increased the initial overshoot by less than 1 F and had an almost negligible effect on the operating economy. Satisfactory performance could be obtained with any set of gains which produced stable control and did not take an excessively long time to adjust to a setpoint change. As a final note it should be repeated that the non-linearities and random disturbances described previously meant that the response was not 100% predictable. It was not uncommon for the morning start-up on two separate days to require almost identical setpoint changes, yet on one day the control would overshoot the setpoint during the first 15 minute period and on the other day it would undershoot the setpoint.

5.3.3 <u>Controller Performance</u> The performance of the microprocessor controller has been described in the preceding paragraphs, a plot of this performance is given in Figure 5.8. This graph shows the calculated optimum percent outside air as well as the percent actually achieved during a weekend in March. As a comparison, the performance of the system when operating on the manual override is also given in Figure 5.8. The data for the

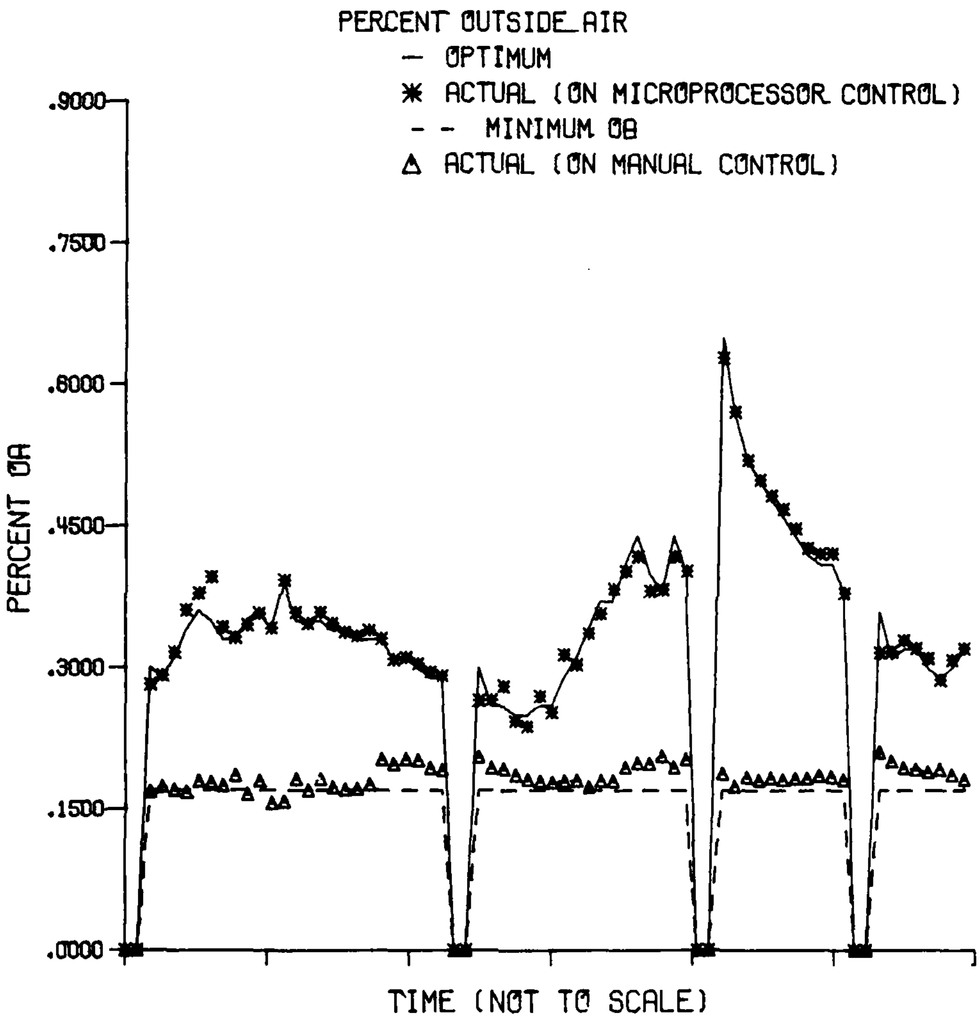


Figure 5.8

Performance Of Controller

microprocessor and manual control modes were, obviously, taken on different weekends. The data for each mode shows the performance over a four day period extending from a Friday morning to the following Monday morning. (System ACP-7 is not normally operated on Sundays, but on these particular weekends unusual occupancy schedules necessitated Sunday operation.) "False" data indicating 0% outside air has been inserted to indicate periods when the system was shut down at night, no data was actually logged while the system was off. Each successive data point was logged 30 minutes after the preceding point, but the periods when the system was shut down have been shortened considerably.

Figure 5.8 shows that the microprocessor control followed the setpoint fairly closely, even when the morning start-up caused a large setpoint change. The performance on manual control shows that fixed dampers provided a relatively constant percent outdoor air, wind gusts and temperature changes did not have much affect on the mixture. The manual positioning adjustment was not touched during this period (providing a constant pressure to the dampers every day), but the dampers apparently opened a little further on Monday than on Sunday. This did not cause any problems or affect the operating economy significantly, and it could be prevented by the common practice of using a mechanical stop to limit the damper travel.

RESULTS

The savings which can be achieved by any economizer cycle are very much dependent on the weather, therefore discussion of the results will begin with a discussion of the weather encountered during the test. An estimate of the savings achieved will follow, and then an analysis of how the load varied with weather and with the time of day will be done to provide a comparison between the demand based economizer and a conventional economizer.

6.1 Weather

The equipment used to implement the microprocessor control scheme was installed during the month of January 1984, and data on its performance were taken during the months of February, March, and April. The weather which occurred during these months is summarized in tables 6.1 through 6.3. These tables show how many hours the system was operating in a minimum outside air mode and in the economizer mode during each 5 degree bin of outside air temperature, together with the total predicted duration of each bin based on weather data gathered by the U.S. Air Force (14). The total number of hours the system was operated varies significantly from the total number of hours predicted primarily because the predicted data is based upon 24 hr/day, 7 day/week observations whereas the microprocessor did not log data when system ACP-7 was shut down at night and on weekends.

Table 6.1 shows that the month of February was considerably warmer than predicted. The actual hours for all bins warmer than the 35 to 39.9 degree bin is greater than the predicted values and one bin, 65 to 69.9 deg, was not even included in the predicted data. The fact that the cooler temperatures were not experienced for as many hours as predicted is partly attributable to the unusually warm weather and partly attributable to the fact that the microprocessor was shut down during the night hours when these cool temperatures were most likely to occur. Since the weather during February was warmer than predicted, it would be expected that the economizer would show greater savings than that predicted by the typical weather data.

Table 6.2, on the other hand, shows that the weather encountered in March was much cooler than that predicted. Again the night shut-down prevented the microprocessor from seeing the coolest temperatures, but the warm weather above 55 F which should have occurred during the daytime (when system ACP-7 was operating) simply did not occur. The fact that the microprocessor ran in the economizer mode for 26 hours more than it ran in the minimum OA mode is primarily coincidence. The economizer was operated on the "one day on, one day off" scheme throughout this month; however, since system ACP-7 was run on an erratic schedule over weekends the economizer mode was not changed between Friday morning and Monday morning. The month of March included five weekends, during two of these the microprocessor was in a "min OA" mode and during three of these it was in the "economizer" mode. Thus the total number of hours in the economizer mode is greater than the total in the min OA mode. Since the weather in March was cooler than predicted, it would be expected that the economizer could not provide the savings predicted by using typical weather data.

Table 6.3 shows that the weather during April was slightly cooler than predicted. No temperatures warmer than 75 F were encountered, and system operated the greatest number of hours in the 40 to 44.9 F temperature bin. The predicted weather showed an almost equal number of hours in each of the bins from 40 F to 60 F, but the actual data shows the number of hours in the 55 to 59.9 F bin to be less than a third of those in the 40 to 44.5 F bin. Since these temperature bins include the range where the economizer should function best, it would be expected that the economizer would not function quite as well as predicted during April. The fact that the warm temperatures above 75 F were not observed should help boost the economizer performance slightly, since these temperatures are too warm to allow the use of the economizer and operation in these bins would increase the total cost without contributing to the savings.

Table 6.1

Bin Analysis Of February Weather

Bin (OA Temp)	Hours On Min. OA	Using Economizer	Total	Predicted
65 to 69.9	1.5	0	1.5	0
60 to 64.9	2.0	2.5	4.5	1.0
55 to 59.9	11.0	4.5	15.5	4.0
50 to 54.9	12.0	9.5	21.5	11.0
45 to 49.9	26.5	20.5	47.0	18.0
40 to 44.9	31.5	42.5	74.0	35.0
35 to 39.9	22.5	37.5	60.0	85.0
30 to 34.9	23.5	16.5	40.0	136.0
25 to 29.9	26.0	22.5	48.5	118.0
20 to 24.9	7.5	3.0	10.5	90.0
15 to 19.9	8.5	3.0	11.5	68.0
10 to 14.9	3.0	9.0	12.0	44.0
5 to 9.9	0.5	1.0	1.5	27.0
0 to 4.9	0	0	0	20.0
-5 to -0.1	0	0	0	10.0
-10 to -5.1	0	0	0	3.0
Total	176.0	172.0	348.0	670.0

Table 6.2

Bin Analysis Of March Weather

Bin (OA Temp)	Hours On Min. OA	Using Economizer	Total	Predicted
75 to 79.9	0	0	0	1.0
70 to 74.9	0	0	0	6.0
65 to 69.9	0	0	0	8.0
60 to 64.9	0	0	0	15.0
55 to 59.9	0	0	0	27.0
50 to 54.9	6.0	1.0	7.0	40.0
45 to 49.9	9.5	13.0	22.5	55.0
40 to 44.9	19.5	26.5	46.0	97.0
35 to 39.9	37.0	44.5	81.5	142.0
30 to 34.9	46.5	43.0	89.5	156.0
25 to 29.9	19.5	33.0	52.5	96.0
20 to 24.9	21.0	23.5	44.5	58.0
15 to 19.9	11.5	9.5	21.0	27.0
10 to 14.9	0	1.0	1.0	11.0
5 to 9.9	0	1.5	1.5	5.0
0 to 4.9	0	0	0	1.0
-5 to -0.1	0	0	0	1.0
Total	170.5	196.5	367.0	746.0

Table 6.3

Bin Analysis Of April Weather

Bin (OA Temp)	Hours On Min. OA	Using Economizer	Total	Predicted
80 to 84.9	0	0	0	6.0
75 to 79.9	0	0	0	15.0
70 to 74.9	10.0	6.0	16.0	26.0
65 to 69.9	4.5	8.0	12.5	44.0
60 to 64.9	11.5	14.5	26.0	71.0
55 to 59.9	10.5	9.5	20.0	97.0
50 to 54.9	34.5	20.5	55.0	97.0
45 to 49.9	33.0	16.5	49.5	102.0
40 to 44.9	24.0	45.5	69.5	103.0
35 to 39.9	5.0	21.0	26.0	86.0
30 to 34.9	0	0	0	52.0
25 to 29.9	0	0	0	17.0
20 to 24.9	0	0	0	2.0
Total	133.0	141.5	274.5	718.0

6.2 Savings

The data logged every 1/2 hour by the microprocessor included the temperature differences across the heating and cooling coils as well as the airflow through each coil, so the operating cost for each coil could have been estimated on a dry bulb temperature basis. This would not have included the cost of any condensation which occurred in the cooling coil, so the cost estimates used to analyze the system performance were based upon a modified version of program Pdeck. In this modified version the actual temperatures and flows measured by the microprocessor were combined with the humidities predicted by the U.S. Air Force Bin Data, and operating costs were calculated based upon the enthalpy changes across the coils. During the months of February through April these costs varied only very slightly from those predicted by a dry bulb temperature analysis, and this finding was supported by the fact that random observations of the cooling coil drain pan showed very little condensation was in fact occurring. These costs were used to estimate the savings achieved by two distinct methods. In one method the costs were combined into an average operating cost per hour for each operating mode and for each "bin" of outside air temperature. The total monthly operating cost for the economizer mode was calculated by multiplying the hours it operated in this mode during any one bin by the average operating cost for that bin and summing the products for all bins. The operating cost which would have occurred if the economizer had not been used was then estimated by multiplying the same operating hours by the minimum OA cost per hour figures and summing those products. The difference between these two totals gave the total savings achieved by the economizer during that month. Implicit in this method was the assumption that the heating and air conditioning costs are primarily determined by the outside air temperature so that an average operating cost for each bin was a reliable basis for comparison.

A second method of estimating costs, one which did not rely upon this assumption, was to use a Pdeck-like analysis to simulate the building. In this method the actual temperatures and flows measured in the economizer mode were used to determine the heating and cooling coil loads, the cost of operating in the economizer mode was calculated from these loads, and the cost of operating in the minimum outside air mode was then estimated based on the assumption that the coil exit conditions (temperatures and flows) would not change as the mixed air temperature changed. This method allowed the building load to vary independently of the outside weather conditions, but did not account for the fact that the offset inherent in the pneumatic deck controllers did in fact allow the coil discharge temperatures to vary as the mixed air temperature varied. This change in coil discharge temperature caused the airflow through the coils to vary as well. The Pdeck-like analysis also assumed that the dampers would provide exactly the correct mixture of outside and return air to meet minimum outside air specifications, whereas Figure 5.8 showed the dampers are not that precise.

The cost calculations for the months of February, March, and April together with the savings estimated by the two methods just described are summarized in Tables 6.4 through 6.6. The cost figures predicted by program Pdeck are also shown in these tables to provide a basis for comparison. The Pdeck figures were based upon 24 hr/day, 7 day/week operation and were calculated using a very rough simulation of the heating and cooling loads, so they are only included to provide a starting point for the analysis.

Table 6.4

Bin Analysis Of February Costs

Bin (OA Temp)	Average Cost ($/hr) On Min OA Act.	On Min OA Pred.	Economizer Act.	Economizer Pred.	Average % Savings Act.	Average % Savings Pred.
65 to 69.9	1.03	-	-	-	-	-
60 to 64.9	0.94	1.19	0.75	0.39	20	67
55 to 59.9	0.96	1.07	0.73	0.37	24	60
50 to 54.9	0.91	0.96	0.72	0.44	21	54
45 to 49.9	0.89	0.86	0.66	0.52	26	26
40 to 44.9	0.83	0.80	0.60	0.60	28	25
35 to 39.9	0.74	0.76	0.57	0.67	23	12
30 to 34.9	0.72	0.73	0.51	-	29	-
25 to 29.9	0.59	0.73	0.62	-	-5*	-
20 to 24.9	0.59	0.75	0.60	-	-2*	-
15 to 19.9	0.60	0.78	0.52	-	13	-
10 to 14.9	0.62	0.85	0.53	-	14	-
5 to 9.9	0.64	0.94	0.58	-	9	-

Predicted Savings (Program Pdeck, 670 operating hrs):
Cost on Min OA - Cost On Economizer
$529 - $499 = $30 or 5.7%

Actual Savings (using Bin Data, 172 operating hrs):
(Min OA Cost) x (Econ Hrs.) - (Econ Cost) x (Econ Hrs.)
$132 - $103 = $29 or 22.0%

Actual Savings (using Pdeck-like analysis, 172 hrs):
$162 - $103 = $59 or 36.4%

* Negative savings caused by malfunctioning hot deck controller. See text for explanation.

Table 6.5

Bin Analysis Of March Costs

Bin (OA Temp)	Average Cost ($/hr) On Min OA Act.	Average Cost ($/hr) On Min OA Pred.	Average Cost ($/hr) Economizer Act.	Average Cost ($/hr) Economizer Pred.	Average % Savings Act.	Average % Savings Pred.
75 to 79.9	-	1.71	-	-	-	
70 to 74.9	-	1.52	-	1.27	-	16
65 to 69.9	-	1.34	-	0.77	-	43
60 to 64.9	-	1.19	-	0.39	-	58
55 to 59.9	-	1.06	-	0.38	-	64
50 to 54.9	0.77	0.94	0.54	0.45	30	46
45 to 49.9	0.75	0.87	0.49	0.52	35	40
40 to 44.9	0.73	0.80	0.48	0.59	34	26
35 to 39.9	0.72	0.76	0.52	0.67	28	12
30 to 34.9	0.67	0.73	0.53	-	21	-
25 to 29.9	0.65	0.73	0.55	-	15	-
20 to 24.9	0.64	0.75	0.60	-	6	-
15 to 19.9	0.63	0.78	0.60	-	5	-
10 to 14.9	-	0.85	0.58	-	-	-
5 to 9.9	-	0.94	0.58	-	-	-

Predicted Savings (Program Pdeck, 746 operating hrs):
Cost on Min OA - Cost On Economizer
$605 - $496 = $109 or 18.0%

Actual Savings (using Bin Data, 194 operating hrs):
(Min OA Cost) x (Econ Hrs.) - (Econ Cost) x (Econ Hrs.)
$133 - $104 = $29 or 21.8%

Actual Savings (using Pdeck-like analysis, 194 hrs):
$155 - $105 = $49 or 31.9%

Table 6.6

Bin Analysis Of April Costs

Bin (OA Temp)	Average Cost ($/hr) On Min OA Act.	Pred.	Economizer Act.	Pred.	Average % Savings Act.	Pred.
80 to 84.9	-	2.01	-	-	-	-
75 to 79.9	-	1.77	-	-	-	-
70 to 74.9	1.30	1.57	1.23	1.27	5.4	19.1
65 to 69.9	1.22	1.35	1.00	0.77	18.0	43.0
60 to 64.9	0.97	1.19	0.60	0.39	38.1	59.3
55 to 59.9	0.86	1.06	0.56	0.38	34.9	64.2
50 to 54.9	0.81	0.96	0.61	0.45	24.7	53.1
45 to 49.9	0.80	0.87	0.54	0.52	32.5	40.2
40 to 44.9	0.79	0.80	0.51	0.59	35.4	26.3
35 to 39.9	0.69	0.76	0.52	0.67	24.6	11.8
30 to 34.9	-	0.73	-	-	-	-
25 to 29.9	-	0.73	-	-	-	-
20 to 24.9	-	0.75	-	-	-	-

Predicted Savings (Program Pdeck, 718 operating hrs):
Cost on Min OA - Cost On Economizer
$708 - $437 = $271 or 38.3%

Actual Savings (using Bin Data, 141.5 operating hrs):
(Min OA Cost) x (Econ Hrs.) - (Econ Cost) x (Econ Hrs.)
$120 - $85 = $35 or 29.2%

Actual Savings (using Pdeck-like analysis, 141.5 hrs):
$140 - $85 = $55 or 39.3%

Table 6.4 shows that, as expected, the economizer performed much better during the month of February than program Pdeck had predicted. The bin by bin analysis shows the economizer was not as effective at higher temperatures as Pdeck had predicted but was much more effective at lower temperatures. The low temperature performance is primarily attributable to the existence of the perimeter heating system, a system which was not simulated by Pdeck. This system drastically reduced the heating load at low temperatures and therefore made economizer operation more desirable. The negative savings shown in the 20-30 F bins are misleading, as they were caused by the previously described problem of low supply air pressure to the pneumatic hot deck controller. When the supply air pressure was first corrected so that the hot deck controller could control the coil, the hot deck temperature dropped well below 75 F. This was because the controller setpoint had been lowered during previous attempts to bring the coil under control, and it took roughly two days before the controller was properly adjusted. During these two days the rooms served by ACP-7 were considerably cooler than was desirable, and the operating costs were correspondingly reduced. By coincidence, these two days comprised over half the total time that ACP-7 was operating in a min OA mode in these two temperature bins, so the min OA costs for these bins are artificially low.

The poor performance of the economizer at the higher temperatures is probably caused by a number of factors. The min OA costs in these bins are almost identical to those predicted by Pdeck, but the economizer costs are much higher. A large part of this is due to the fact that the economizer system was still being adjusted and balanced when the warm weather occurred, and the economizer could only admit up to 45% outside air instead of the 100% allowed by Pdeck or the 84% which the economizer could admit after the adjustments were completed. The offset problem with the cold deck also hampered the economizer efficiency. Since the mixed air temperature was generally around 3 degrees warmer than the cold deck temperature, the cold deck costs could not be dropped to zero as in Pdeck. The warm February weather occurred before the hot deck had been brought under control, so the hot deck temperature varied between 85 and 95 F. This led to increased hot deck costs in both operating modes. The control system on the perimeter heaters should have caused them to shut down during the warm weather, but it is possible that this system still added to the overall cooling load. This would make economizer operation more desirable, but since the economizer itself was hobbled by the factors already described it would have added to the cooling costs in both modes.

The total savings predicted by the Pdeck-like analysis are roughly 15% greater than those predicted by the bin analysis. The economizer costs for both methods are identical, so the difference lies in the min OA costs. The Pdeck-like analysis assumed the deck temperatures would not be affected by a change in the percent OA admitted, but observations made on the actual system show this was not true. When the system was operating in the economizer mode the lower mixed air temperature generally dropped the cold deck temperature 2 to 4 degrees below the corresponding temperature in the min OA mode. Thus the Pdeck-like analysis was based upon an artificially low cold deck temperature and the calculated min OA costs are too high as a result. The hot deck temperature was also lowered when more than the minimum amount of outside air was admitted, but since the system was primarily operating in a cooling mode the effects of the hot deck offset are not as noticeable as the effects of the cold deck offset. This offset would not be a problem if the decks were controlled by a properly functioning PID controller, so the total savings which could be achieved if the entire system were converted to DDC would probably fall between the 22% predicted by the bin method and the 36% predicted by the Pdeck-like analysis.

Table 6.5 shows the March data supports the conclusions drawn from the February data. The warm weather which occurred in February did not occur in March, but the general trend still appears to be that the economizer functioned better than predicted in cold weather and worse than predicted in warm weather. Even though the weather during March was much colder than that which the predicted savings were based on, the total savings during March were a little better than predicted. Again the Pdeck-like analysis showed savings slightly higher than the bin analysis, about 10% higher in this instance. The cold deck offset problem is less severe during cool weather (when the difference between the mixed air temperature on min OA and in the economizer mode is less pronounced), so it is to be expected that the Pdeck-like savings would differ from the bin method savings by a lesser amount during the abnormally cool March than it did during the unusually warm February.

Table 6.6 again shows the economizer functioned better during the cool weather below 45 F than it did during the warmer weather. Since the April weather was warmer than 45 F for roughly 2/3 of the hours that ACP-7 was operating, the system did not save quite as much as program Pdeck had predicted. The 30 to 40% it did save is certainly worthwhile, however. In addition to the limit on the maximum percent outdoor air and the cold deck offset problems already discussed, other factors which limited the savings available during April were the shut-down of the perimeter heating system, a gradual reduction in the hot deck temperature, and a slight lowering of the chilled water temperature being supplied to the cooling coil. The first two changes reduced the air conditioning load on the system and therefore reduced the potential for economizer savings, and the third change caused the cold deck to run between 3 and 5 F cooler than it had during the winter. Since the microprocessor assumed the cold deck setpoint was fixed at 63 F and did not adjust to the new setpoint, the cooling costs were increased by this change in the chilled water temperature. Again, this would not have been a problem if the microprocessor had controlled the cold deck as well as the economizer. The perimeter heating system shut-down and the drop in the chilled water temperature are changes which are made campus wide every spring as part of the switch-over from a winter heating mode to a summer cooling mode. The drop in the hot deck temperature is a result of various adjustments made to help bring the heating coil under better control.

The data taken during the three test months indicates the economizer works at least as well as the Pdeck simulation predicted. If the test were extended over an entire year the savings during the summer months might not be quite as great as Pdeck predicted, but the savings during the fall and winter would probably be greater. Thus it does not seem unreasonable to assume a total annual savings of at least 22% or $2187, as predicted by Pdeck. The total cost of the microcomputers, sensors, and associated electronics used in this experiment was approximately $2450, so experimental system could pay for itself in a little over a year. No attempt was made to minimize equipment costs for this experiment, a system could be built to perform the same function for much less. If a 9 channel A/D board had been used instead of two 8 channel boards, for example, the system could be made to operate on a single microcomputer and a single A/D board, which would reduce the equipment cost by $825. Total costs for designing, purchasing, and installing commercially available DDC systems are commonly estimated at between $300 and $400 per point, so the 9 input / 1 output system used in this experiment would cost between $3000 and $4000. This is a "worst case" cost estimate, it is highly unlikely that anyone would be interested in purchasing a complete DDC unit just to control an economizer. A better strategy would be to use a DDC unit to control the entire HVAC system and to include the economizer program as one of several energy saving routines. If a suitable DDC unit had already been installed on system ACP-7, for example, the sensors, A/D cards, and E/P transducer needed to implement the demand based economizer routine would have cost an additional $1060. If the existing DDC unit included a conventional economizer so that the temperature sensors and E/P transducer were already in place, the equipment needed to convert to a demand controlled economizer would have cost an additional $350. Obviously there are many ways to estimate costs, the figures given here indicate the payback period for a demand controlled economizer ranges from 6 months (if the economizer is added to an existing DDC unit) to 2 years (if a DDC unit is purchased just for this purpose). In either case, the payback period falls well within the range which is normally considered to be economically justifiable.

6.3 Load Variations

The discussion thus far has compared the operating costs using the demand based economizer to the costs using minimum outside air. Some of the savings observed could also have been achieved by a conventional economizer which based the control decision on the outside air temperature. It is difficult to provide a quantitative comparison between the savings which could be achieved by the two types of economizers because the performance of the conventional economizer is dependent upon what temperature is chosen as the lower limit as well as on the weather conditions described so far. A qualitative feel for how a conventional economizer would have performed in system ACP-7 can be gained by looking at how the heating and cooling loads varied with outside air temperature. This is plotted in Figure 6.1. The heating and cooling loads are plotted as net cooling cost (= cold deck cost - hot deck cost) to relate these loads to economizer performance. When the net cooling cost is positive, it indicates system ACP-7 was operating in an air conditioning mode and an economizer would cut operating costs. The predictions done by program Pdeck (as well as the performance of the microprocessor) show that even if the net cooling cost is negative the total system cost may still be lowered somewhat by switching to an economizer mode, but the savings achieved are not nearly as great as when the system cooling cost is positive. Thus the dashed line in Figure 6.1 (net cooling cost = 0) gives some indication of the lower limit for effective economizer operation. The data plotted on this graph were taken when system ACP-7 was operating in the minimum OA mode only, as the net cooling cost in the economizer mode would be greatly reduced by the economizer and the "cost = 0" operating line would have little meaning. All data on this graph were taken during the month of March.

Figure 6.1 shows economizer operation was always feasible above a temperature of about 32 F, and was sometimes advisable at temperatures below this point. Thus a conventional economizer with a low temperature cut-off of, say 30 F, could have achieved much of the savings achieved

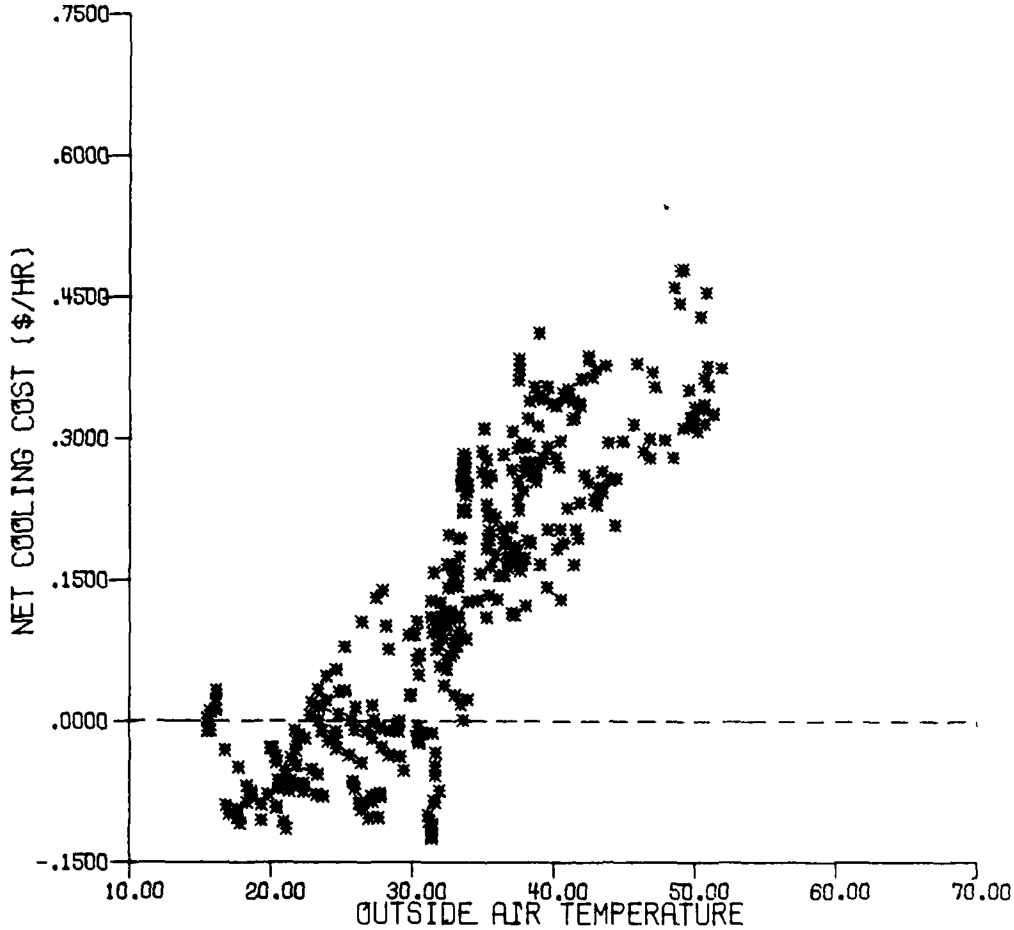


Figure 6.1

Cooling Cost Vs Outside Air Temp

by the demand based economizer. In the region below 32 F the demand based economizer would be able to outperform a conventional economizer, but the savings to be achieved in this region are not as great as those available at higher outside air temperatures. Figure 6.1 does show that although there is a general increase in net cooling cost as the outside air temperature increases, there can be considerable variation in load at any one temperature. In the region of 32 to 35 F, for example, the cooling cost varies from + $.30/hr to - $.12/hr. Obviously the outside air temperature only provides a general indication of what the cooling load is. There will always be a 15-20 degree "range of uncertainty" in which a conventional economizer will not always make the correct decision. Also it should be noted that the Figure 6.1 is based upon flow and temperature readings taken by the demand based economizer. If this instrumentation had not been installed in the air conditioning system it would have been impossible to determine where to set the low limit for a conventional economizer. Typically this cut-off is set around 45-50 F. Figure 6.1 shows that if this had been done with system ACP-7 a conventional economizer would have missed virtually all of the potential savings. Program Pdeck predicted the optimum cut-off would be near 35 F. A conventional economizer with this as a low limit would have achieved over half the savings available, but would still have missed a great deal. The demand based economizer, however, was able to realize savings at temperatures as low as 5 F.

Another point which should be made is that Figure 6.1 shows the net cooling cost for system ACP-7 as it was operating in March of 1984. If the low temperature cut-off of a conventional economizer were set based upon this plot, it would only be correctly adjusted for as long as the operating conditions remained unchanged. If the set-point on the perimeter heating system were changed, for example, the heating and cooling loads on system ACP-7 would change and Figure 6.1 would no longer be valid. The demand based economizer will autcmatically adjust to any changes in the building load, but a conventional economizer will not. In short, it may be possible to adjust a conventional economizer so that it will achieve much of the savings available to a demand based economizer, but this adjustment will require repeated temperature and flow measurements and will need to be repeated whenever any element in the building system is changed if peak performance is to be maintained.

A plot of how the net cooling cost varies with the time of day is shown in Figure 6.2. This graph shows the net cooling costs in the min OA mode on Mondays, Wednesdays, and Fridays during the month of March. (These three

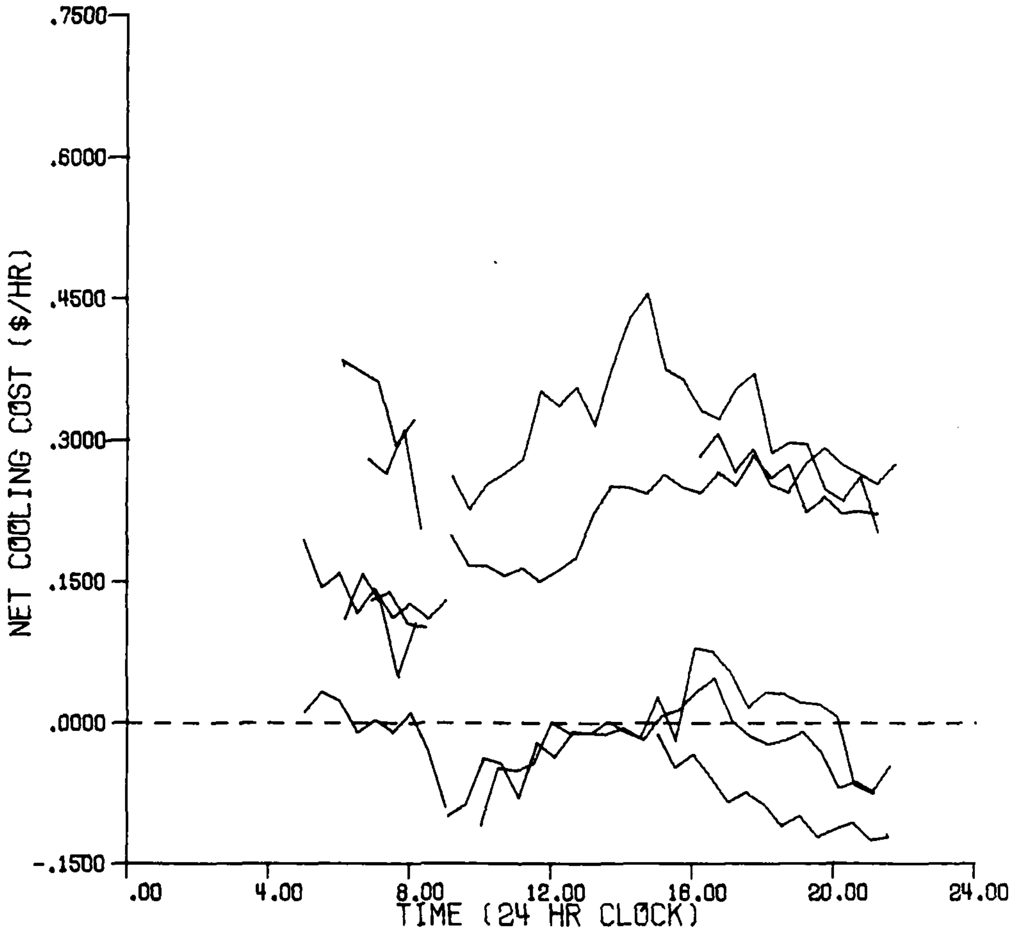


Figure 6.2

Cooling Cost Vs. Time Of Day

days were chosen because class schedules are generally identical on these days, so the building occupancy schedules should also be identical.) Each line segment represents a continuous string of measurements during one day. The discontinuities indicate times when the system was switched into the min OA mode from the economizer mode or vice versa. Generally this was done between 8 and 9 am. The absolute magnitude of any one line is relatively meaningless, as this is determined by that day's weather as much as anything else, but the variation throughout the day is interesting. When the system is first started in the morning the load is typically at or very near its peak air conditioning load. Apparently the perimeter heating system, which runs all night long, is heating the building to the point where air conditioning is required to cool it back down to setpoint. The load steadily drops until 8 or 9 am, partly because the system has cooled the building from its nightime high and partly because people arriving for work are opening doors and admitting cold air. (There are no exterior doors in the area served by system ACP-7, so the effect of this entering air would be limited to that caused by air circulation within the building.) The cooling load rises throughout the day until it peaks at around 3 or 4 pm, and then begins dropping as the people leave and as the outside air temperature drops. It is impossible to tell how much of this load variation is caused by outside temperature changes (which would be detected by a conventional economizer) and how much is caused by internal load changes (which would not). The fact that the peak cooling load often occurs early in the morning when the outside air temperature should be near its minimum indicates the internal load can affect the cooling cost as much or more than daily temperature fluctuations It is this fluctuation of the internal load which makes the demand based economizer superior to a perfectly tuned conventional economizer. The more this load fluctuates, the more desirable the demand based economizer becomes.

SUMMARY AND CONCLUSIONS

The data presented in the previous chapters clearly indicate the fact that the demand based economizer is a viable control scheme. During the months of February and March this controller cut the total heating and cooling coil costs in system ACP-7 by well over 20% (compared to minimum outside air operation), and data taken during April indicate such a controller could cut costs by 30% or more under favorable conditions (Savings of over 60% were recorded for some individual days.) Clearly there is great potential for improved operating efficiency if such a scheme is adopted. The fact that the system will pay for itself in less than 2 years (perhaps even in 6 months) adds further justification for its use.

The superiority of the demand based economizer over a conventional economizer is not as easily quantified, but the data did indicate areas in which the demand based economizer would outperform even a perfectly tuned conventional economizer. The phrase "perfectly tuned" is crucial to this comparison - an improperly adjusted conventional economizer would not reduce operating costs nearly as much as the demand based economizer. Since a conventional economizer can only be perfectly adjusted if a complete record of temperatures and flows is available, the same sensors and instrumentation would be required for both types of economizers. In reality, these measurements are not used to adjust conventional economizers and it is doubtful if any "perfectly tuned" economizers actually exist. The demand based economizer will always adjust to the current operating conditions, so it is safe to assume that it will outperform existing conventional economizers. Even if a perfectly tuned conventional economizer did exist, a demand based economizer would outperform it if the internal building load varied independently of the weather. The building chosen as a test site for this experiment showed a moderate fluctuation in internal load; if this fluctuation had been larger the demand based economizer would have shown even greater savings.

The equipment used in this experiment functioned very well throughout the test period. This equipment was not built to the durability and convenience standards required of commercial HVAC controls, but the fact that the "bread-board" system operated so well indicates there should be no major problems in building a commercial system to implement the same control scheme. The heated thermistor velocity sensors showed particular promise, as they may prove to be less expensive than the flow sensors currently on the market.

As a final note, the instrumentation installed for use in this experiment provided invaluable data on how the entire HVAC system was operating, data which allowed the existing controls to be "fine tuned" for better efficiency. The flow sensors were particularly valuable, as they provided information which is not ordinarily available with conventional control systems. As an example, the perimeter heating system in the test area provided so much heat that system ACP-7 was in an air conditioning mode even during the coldest weather in January. This is very inefficient, the cooling coils in system ACP-7 and the perimeter heating system were "fighting" each other, but without the flow sensors this condition would have gone undetected. The rooms remained comfortable and the deck temperatures remained within their normal operating range, so temperature sensors alone would not have detected this problem. The flow sensors; however, indicated that most of the airflow was being channeled through the cooling coil, a situation which is distinctly abnormal during subzero weather. There were many other instances in which the data logged by the microprocessor gave valuable insight into how the existing controls were functioning, which points out the need for proper instrumentation in any type of control system.

RECOMMENDATIONS

The results obtained during this experiment show that development of a commercial version of the demand based economizer would be very desirable. A longer term test of, say, 1 year would help determine the economic feasibility of such a control, but the results obtained to date may be sufficient to justify concurrent development and testing programs. This work is perhaps best left to the established control manufacturers, although it is entirely possible that existing DDC control systems (such as the one currently being installed at Purdue) could be modified to implement a form of this control scheme.

The heated thermistor velocity sensors used in this experiment showed great promise, but a longer term and larger scale test would be required to determine their suitability for operational use. It appears that some controls manufacturers are already working on this type of sensor, so any future work in this field should begin with a survey of manufacturers. Even if the demand based economizer control scheme is not adopted, further work on inexpensive flow sensors is recommended. The data gathered during this experiment showed that flow measurements can give invaluable insight into how a HVAC system is operating and can be used to "fine tune" the entire control system.

In addition to the development of a better velocity sensor, further work should be done on the relationship between the centerline velocity and the average velocity in rectangular ductwork. The assumption that these two velocities are related by a constant ratio of 1:0.9 served the needs of this experiment, but is not recommended for wider use until further study is done. It is also possible that further study could reveal a better location to take the velocity measurements. In round ducts, for example, a velocity probe located at a radius of 0.762 R (R = radius) from the centerline will indicate the average velocity to within $\pm$ 1/2 percent over a wide range of Reynold's numbers (28). If a similar relationship could be found for non-circular ducts the accuracy of the flow measurements could be greatly improved.

The outdoor air temperature sensor used in this experiment functioned very well after it was lengthened to reduce conduction problems (see chapter 5), but its performance before this modification was very poor indeed. Clearly this could be a problem with other sensors as well. Some point sensors with long support brackets are now appearing on the market. It is recommended that all future sensing of outdoor air temperatures be done with one of these sensors, with an averaging sensor, or with some other type of sensor which will eliminate the conduction problem. It is also recommended that outdoor air temperatures be measured in ductwork leading to the system being controlled, or at least in ductwork leading to a similar system in the same building. Some large-scale energy management systems have proposed using a single outdoor air sensor to control economizers located throughout a campus, but temperature measurements made in connection with this experiment indicate local variations in outdoor air temperature can have a significant effect on economizer performance. These measurements were not actually a part of the experiment and were not conducted in a rigorous manner, but until further experiments are done the use of a single sensor to control several buildings is not recommended.

The instrumentation installed as part of this experiment revealed several minor problems with the existing controls on system ACP-7, and it is quite possible that many of the "fine tuning" adjustments made to this system may be applicable to other similar systems. Further measurements are recommended to determine how widespread the problems noted in this report are. Clearly there is a possibility that other perimeter heating systems may be set too high. Hand held flow meters could be used to measure the hot and cold deck flows on a cold day, if the rooms are calling for more cold air than warm air the perimeter heating system is probably set too high. Many of the conventional economizers which were disabled in the past may need to be reactivated. A building by building analysis using current energy costs is therefore recommended. Again, a hand held flow meter could be used to measure the deck flows to prepare such an analysis and to set the low temperature cut-off on any economizers which are re-activated.

As a final note, it should again be stressed that system ACP-7 has been very well maintained and was in fine operating condition at the beginning of this test. Many existing HVAC systems are not in nearly as good condition, and the savings achieved by the demand based economizer are insignificant compared to the savings which can be obtained by repairing a malfunctioning control system. The first step which should be taken in any energy management program is to get the existing systems to operate as designed. Only after this is done should more complicated control schemes, such as the one described in this thesis, be considered.

BIBLIOGRAPHY

BIBLIOGRAPHY

1. Karnon, Philip A., Outlook ´83, ASHRAE Journal, pp. 24-27, February 1983

2. Energy Research And Development Agency (ERDA), Overview And Recommendations, Technical Opportunities For Energy Conservation In Buildings Through Improved Controls, Conference, May 10, 1976

3. Welbourn, John A., and Pauley, John H., A Low Technology Approach To Energy Conservation In Existing Buildings, Herrick Laboratory Conference On HVAC Improvements, Purdue University, 1978

4. Shoenberger, Paul K., Energy Saving Techniques Applied To HVAC Systems In Commercial Buildings, Herrick Laboratory Conference On HVAC Improvements, Purdue University, 1978

5. American Society Of Heating, Refrigeration, And Air Conditioning Engineers (ASHRAE), Handbook Of Fundamentals (1981), pg 22.1

6. Gerhold, R. J., Test And Analysis Of Automatic Damper Operation In HVAC Systems For Maximum Energy Conservation, Herrick Laboratory Conference On HVAC Improvements, Purdue University, 1978

7. Johnson Controls, 507 E Michigan St, P.O. Box 423, Milwaukee, Wis. 53201, Energy Conservation Control (DSC 8500 Manual), pg. 3, no date available

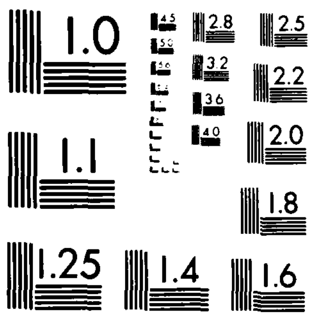
8. Nestor, D.W., The Economizer Cycle: Precis And Suggested Improvements To Existing HVAC Systems, Proceedings Of The First National Conference On Technology For Energy Conservation, Rockwell Md, pp. 70-76, 8-10 June, 1977

9. McKew, Howard J., Double Duct - A Better Way, Herrick Laboratory Conference On HVAC Improvements, pp. 237-241, 1978

AD-A144 136 DEMAND CONTROLLED ECONOMIZER CYCLES: A DIRECT DIGITAL CONTROL SCHEME FOR..(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH S T TOM MAY 84 3/3

UNCLASSIFIED AFIT/CI/NR-84-33D F/G 13/1 NL

END
DATE
FILMED
9-84
DTIC



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963-A

10. Lambert, Peter C., and Engineer, P., Conversion To Dynamic Operation Of Existing Buildings By Using Simple Economic Controls, ASHRAE Transactions, Transaction # CI 81-1 No. 2, 1981

11. American Society Of Heating, Refrigeration, And Air-Conditioning Engineers (ASHRAE), Ventilation For Acceptable Indoor Air Quality (ASHRAE Standard #62-1981), 1791 Tullie Circle NE, Atlanta, Ga. 30329

12. Janeke, C.E., Free Cooling: A Total HVAC Design Concept, ASHRAE Transactions, Transaction # 2664, 1982

13. Kallen, Howard P., Analysis: Off-Peak Cooling Methods To Reduce Energy Consumption, ASHRAE Journal, pp. 30-33, December, 1982

14. United States Air Force, Air Force Manual 88-29: Engineering Weather Data, pp. 3-136 and 3-137

15. Hittle, Douglas C., Dolan, William H., Leverene, Donald J., and Rundus, Richard, Theory Meets Practice In A Full Scale HVAC Laboratory, ASHRAE Journal, pp 36-41, November 1982

16. Johnson Controls, 507 E Michigan St, P.O. Box 423, Milwaukee, Wis. 53201, Energy Conservation Control (DSC 8500 Manual), no date available

17. Hole, V.H.R., Velocity Of Air Measured By Thermistor, Electronic Engineering, pp. 13-15, September 1974

18. Mayhill, Terrence D., and Miller, Paul L. Jr., A Simplified Method For The Measurement Of Air Flow Rates In Flat Oval Ducts, ASHRAE Transactions, Transaction # 2252, 1972

19. Ahmed, S., and Brundrett, E., Turbulent Flow In Non-Circular Ducts. Part 1. Mean Flow Properties In The Developing Region Of A Square Duct, Journal Of Heat And Mass Transfer, Vol. 14, pp. 365-375, 1971

20. Cannon, Don L., Fundamentals Of Microcomputer Design: System Hardware And Software, Texas Instruments, 1982

21. Goode, George & Associates, Introduction To Microprocessors: Hardware and Software, Texas Instruments, 1979

22. Texas Instruments, TM 990/U89 Microcomputer User's Guide, Texas Instruments, 1981

23. Kao, James Y., and Pierce, E. Thomas, Sensor Errors: Their Effect On Building Energy Consumption, ASHRAE Journal, pp. 42-45, December 1983

24. Coughlin, Robert J., Dynamic Response Through Computer Control, ASHRAE Transactions, Transaction # CI 81-1 No. 3, 1981

25. Sams, Geoffrey A., Mellor, Roger, and Fielden, Christopher J., The Design Of Direct Digital Control Building Management Systems, ASHRAE Transactions, Transaction # LA 80-10 No. 1, 1980

26. Jones, Arthur E., Distributed Processing And DDC: A Retrofit Application, ASHRAE Transactions, Transaction # LA 80-10 No. 2, 1980

27. Johnson Controls, 507 E Michigan St, P.O. Box 423, Milwaukee, Wis. 53201, Control Theory (Johnson Control Manual), pp. 6-7, no date available

28. Coxon, W.F., Flow Measurement And Control, Heywood & Co. Ltd, London, 1959, pp. 294

General References

Anderson, Norman A., Control Modes By Step Analysis, Instruments and Control Systems, December 1971

Benton, Ronald, Heat Pump Setback: Computer Prediction And Field Test Verification With Improved Controls, ASHRAE Journal, pp. 23-29, December 1982

Bibbero, Robert J., Microprocessors In Instruments And Control, John Wiley & Sons, 1977

Biehl, R.A., Basics Of Computer/Control Device Interfacing, ASHRAE Transactions, Transaction # CH 81-7 No. 3, 1981

Black, Albert W. III, An Introduction To Micocomputers, ASHRAE Transactions, Transaction # CH 81-1 No. 1, 1981

Borden, William Jr., Inexpensive Transducers for The TRS-80, Byte, pp. 416-442, November 1982

Borkey, John M., Distributed Processing Architectures For Computerized EMS, ASHRAE Transactions, Transaction # CH 81-7 No. 1, 1981

Borresen, Bent A., and Hurley, C.W., Direct Digital Control Of A Pneumatically Actuated Air Handling Unit, ASHRAE Transactions, Transaction # TO 82-6 No. 4, 1982

Borresen, Bent A., HVAC Control Process Simulation, ASHRAE Transactions, Transaction # CI 81-9 No. 1, 1981

Bridges, Frand, Controls Of Systems In Buildings, 1978 Herrick Lab Conference On HVAC Improvements, Purdue University, pp. 207-208, 1978

Bruning, Steven F, A Review Of Minicomputer Systems, ASHRAE Transactions, Transaction # CH 81-1 No. 2, 1981

Chapman, William F., Microcomputers Hail New Era In Controls, ASHRAE Journal, pp. 38-42, July 1980

Comber, James, Automatic Process Control Systems (Basic Types, Principles, And Functions Of), Plant Engineering, pp. 191-194, Nov. 15 1979

Department Of Energy (US Government), Basic Research In Engineering: Process And Systems Dynamics And Control, procedings of a workshop conducted during February 1980

Dressel, L.J., Operator Interface Techniques Must Keep Pace With BAS Sophistication, ASHRAE Journal, pp. 29-32, July 1982

Edwards, Harry J., Automatic Controls For Heating And Air Conditioning, McGraw Hill, 1980

Farris, Donald R., and McDonald, Thomas E., Adaptive Optimal Control: An Algorithm For Direct Digital Control, ASHRAE Transactions, Transaction # LA 80-10 No. 3, 1980

Ford, James W., A Direct Digital Control System In Cornwall Ontario, ASHRAE Transactions, Transaction # LA 80-10 No. 4, 1980

Fuhr, David R., Enhancement Of Field Data Control/Monitor Units in Building Automation Systems Through Use Of Single Chip Microprocessors, ASHRAE Transactions, Transaction # CH 81-7 No. 2, 1981

Fullman, Carl, and Helm, Bob M, Dead Band Thermostats - Pro And Con, ASHRAE Journal, pp. 51-53, July 1981

Gartner, J.R., and Harrison, H.L., Dynamic Characteristics Of Water-To-Air Crossflow Heat Exchangers, ASHRAE Transactions, Transaction # 1932, 1965

Guntermann, Alfred E., Energy Management Systemsss: Are They Cost Effective, Heating, Piping, and Air Conditioning, pp. 102-116, September 1982

Haines, John E., Automatic Control Of Heating And Air Conditioning, McGraw Hill, 1961

Haines, Roger W., Controlling For Energy Conservation In HVAC Systems, 1978 Herrick Lab Conference On HVAC Improvements, Purdue University, pp. 209-211, 1978

Haines, Roger W., Control Systems For Heating, Ventilating, And Air Conditioning, 2nd Edition, Van Nostrand Reinhold Co., 1977

Haines, Roger W., Interfacing An EMCS To An HVAC System, Heating, Piping, and Air Conditioning, pp. 109-111, May 1982

Hamilton, Douglas C., Leonard, Robert, and Pearson, Joseph T., Dynamic Response Characteristics Of A Discharge Air Temperature Control System At Near Full and Part Heating Load, ASHRAE Transactions, Transaction # 2303, 1974

Harmon, Kermit S. Jr., Advanced Control Strategies For Energy Conservation In Building Operation, ASHRAE Journal, pp 55-57, July 1981

Jakobczyk, Jack S., Direct Digital Control For VAV Terminal, Heating, Piping, and Air Conditioning, pp. 77-81, February 82

Jud, Henry G., Analyzing Time Response In Control Systems, Plant Engineering, pp. 124-127, Dec. 1967

Jud, Henry G., Applying Control System Basics, Plant Engineering, pp. 137-139, Nov. 1967

Jud, Henry G., Study Of Frequency Response In Control Systems, Plant Engineering, pp. 90-92, January 1968

Kusuda, Tamomi, and Sud, Ish, Update: ASHRAE TC 4.7 Simplified Energy Analysis Procedure, ASHRAE Journal, pp. 33-39, July 1982

Lawrence, John A., Indroducing EMS Into Smaller Facilities, Energy Management, pp. 21-24, May 1982

Letherman, K.M., Automatic Controls For Heating And Air-conditioning, Pergamon Press, 1981

Levine, M. and Moll, L.W., Beyond Setback: Energy Efficiency Through Adaptive Control, ASHRAE Journal, pp. 37-39, July 1981

Lomers, Guillermo B., Direct Digital Control Of HVAC And Other Processes: 1. An Introduction To Central Monitoring And Control Systems, ASHRAE Transactions, Transaction # CH 81-4 No. 1, 1981

McNall, P.E. Jr., A Control Manufacturers View Of Research And Development Needs For Energy Conservation, ERDA Conference On Energy Conservation Through Control, pp. 13-28, 1976

Oglesby, David R., and Green, Walter L., Dynamic Modeling And Simulation Of A Dual-Duct Air Handling System, Proceedings Of The Ninth Annual Pittsburg Conference On Modeling And Simulation, Vol. 9, Part 1, pp. 89-94, 27-28 April 1978 Owens, George R., EMS Part 2: Specifying Your System, How To Get More Than A Glorified Timeclock,Energy Management, pp. 29-33, March 1983

Pedersen, Curtis O., On The Use Of Large Mainframe Computers, Timeshared Networks, & Terminals, ASHRAE Transactions Transaction # CH 81-1 No. 3, 1981

Ranieri, Michael A., Microprocessors In Control Systems, Heating, Piping, and Air Conditioning, pp. 39-44, August 1982

Sander, D., and Dumouchel, P., Room Simulation Algorithms For Use In HVAC Systems Simulation Computer Program, 3rd International Symposium On The Use Of Computersss For Environmental Engineering Related To Buildings, pp. 31-42, 10-12 May 1978

Schneider, Raymond K., HVAC Control System, John Wiley & Sons, 1981

Shavit, G., and Brandt, S.G., Dynamic Performance Of A Discharge Air Temperature System With A P-I Controller, ASHRAE Journal, pp. 37-41, September 1982 Shih, James Y., Control Systems And Principles, ERDA Conference On Energy Conservation Through Controls, pp. 49-96, 1976

Shinskey, Francis G., Energy Conservation Through Control, Academic Press, 1978

Shinskey, Greg, Rules Of Thumb For Adjusting Controllers,Instruments and Control Systems, pp. 11, December 1971

Silverthorne, Paul N., Energy Management Central Control And Monitoring Systems, ASHRAE Journal, pp. 40-42, July 1981

Spethmann, Donald H., Electrical Energy Management, ASHRAE Journal, pp. 31-35, July 1981

Stoecker, W.F., and Daber, R.P, Conserving Energy In Dual-Duct Systems By Reducing The Throttling Ranges Of Air-Temperature Controllers, ASHRAE Transactions, Transaction # 2470, 1978

Stoecker, W.F., and Rosario, L.A, Heidenreich, M.E., and Phelan, T.R., Stability Of An Air-Temperature Control Loop, ASHRAE Transactions, Transaction # 2471, 1978

Swim, William B., Flow Losses In Rectangular Ducts Lined With Fiberglass, ASHRAE Transactions, Transaction # 2505 RP 176, 1978

Thompson, J. Garth, and Chen, Paul N.T., Digital Simulation Of The Effect Of Room And Control System Dynamics On Energy Consumption, ASHRAE Transactions, Transaction # 2541 RP-212, 1979

Thompson, J. Garth, The Effect Of Room And Control System Dynamics On Energy Consumption, ASHRAE Transactions, Transaction # CI 81-9 No. 2, 1981

Tobias, James R., Simplified Transfer Function For Temperature Response Of Fluids Flowing Through Coils, Pipes, Or Ducts, ASHRAE Transactions, Transaction # 2277, 1973

Waddell, Ernest J., EMS Cost Effectiveness: Another Perspective, AHRAE Journal, pp. 35, November 1982

Weaver, Terry R., Optimum Control Of HVAC Systems, Second Annual Proceedings Of The Energy Council Conference On Energy, University Of Missouri At Rolla, pp. 401-403A, October 7-9, 1975

Westphal, Bruce D., Computerized Energy Management: Three Key Software Concepts, Specifying Engineer, pp. 58-62, August 1981

Williams, V.A., Better Control Through Computers, ASHRAE Journal, pp. 17-20, July 1982

Zermuehlen, R.O., and Harrixon, H.L., Room Temperature Response To A Sudden Heat Disturbance Input, ASHRAE Transactions, Transaction # 1935, 1965

APPENDICES

Appendix A: Drawings Of Krannert Building

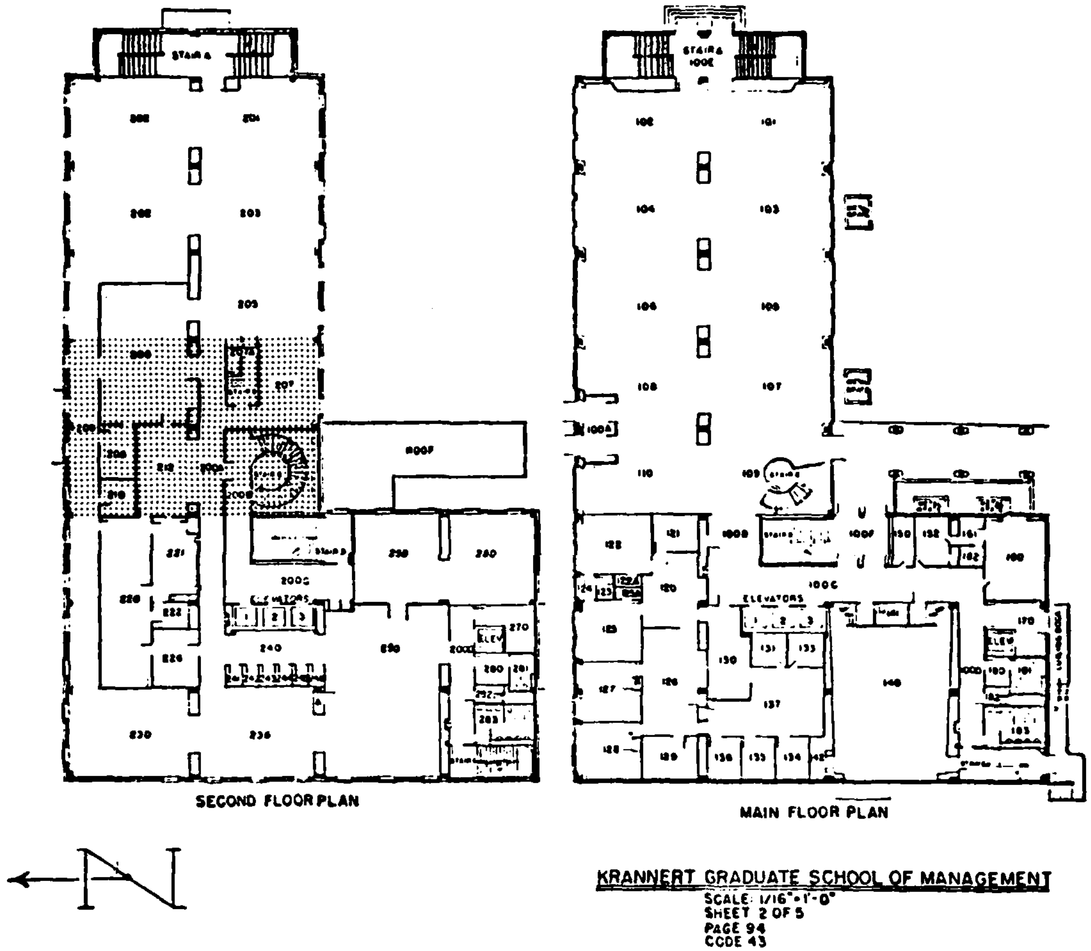


Figure A.1

Krannert Floor Plans: Main And Second Floors

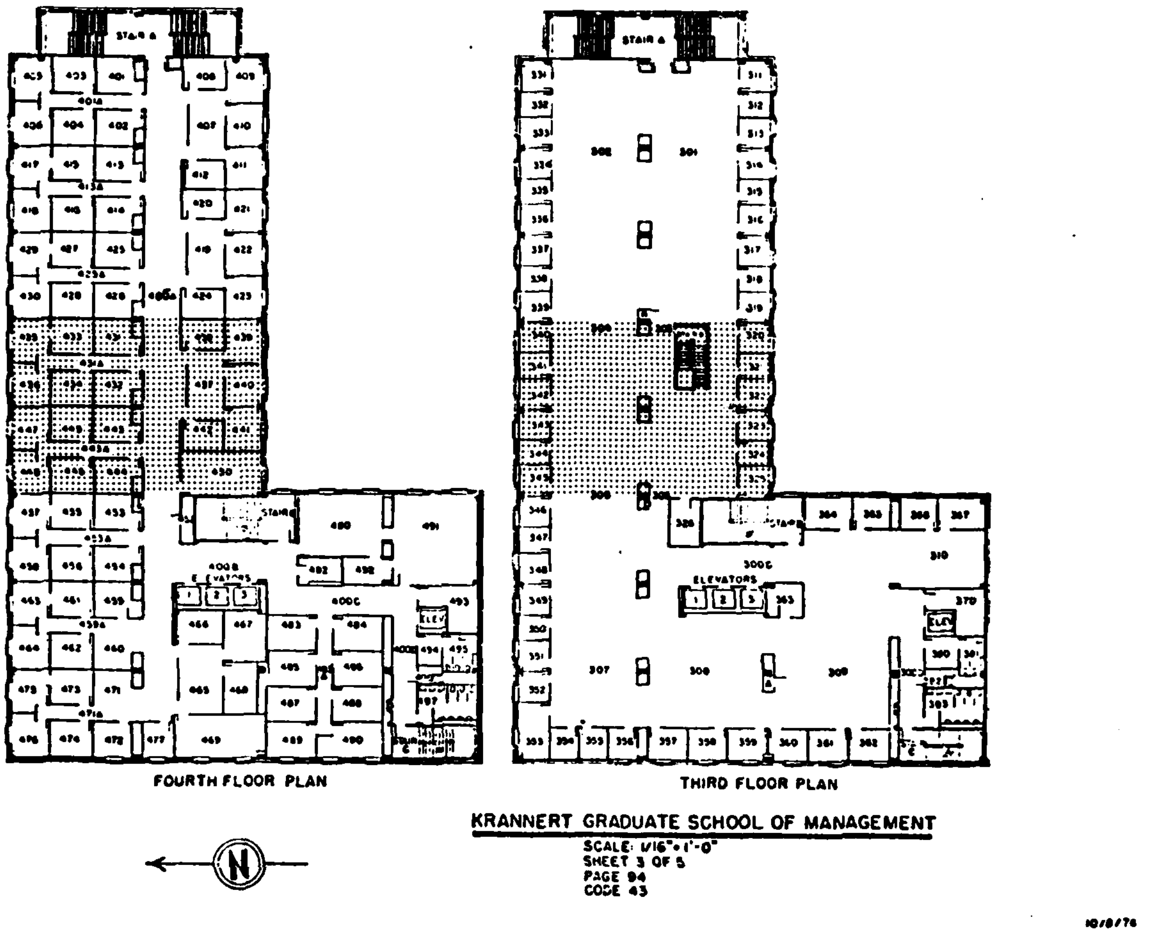


Figure A.2

Krannert Floor Plans: Third A[illegible] Floors

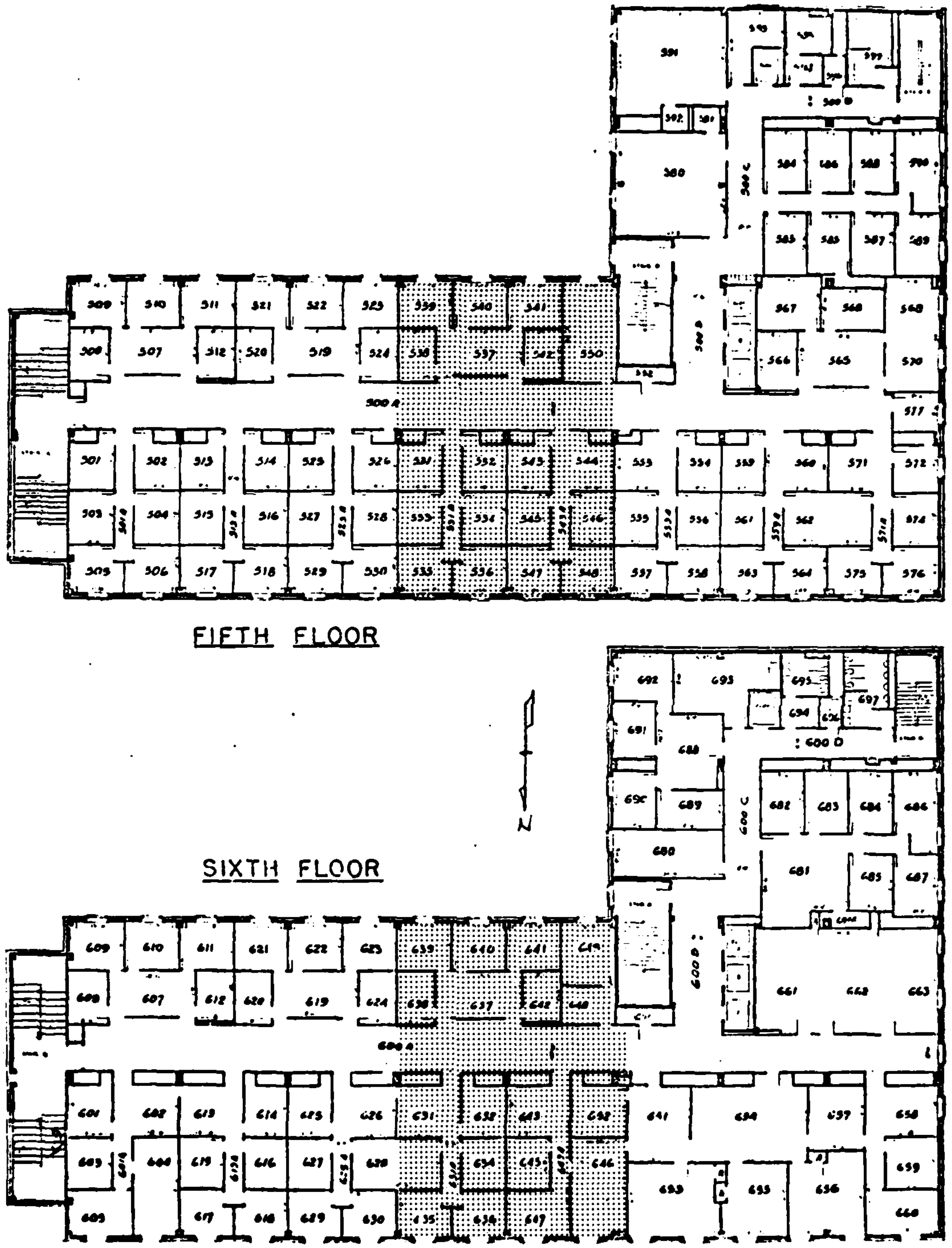


Figure A.3

Krannert Floor Plans: Fifth And Sixth Floors

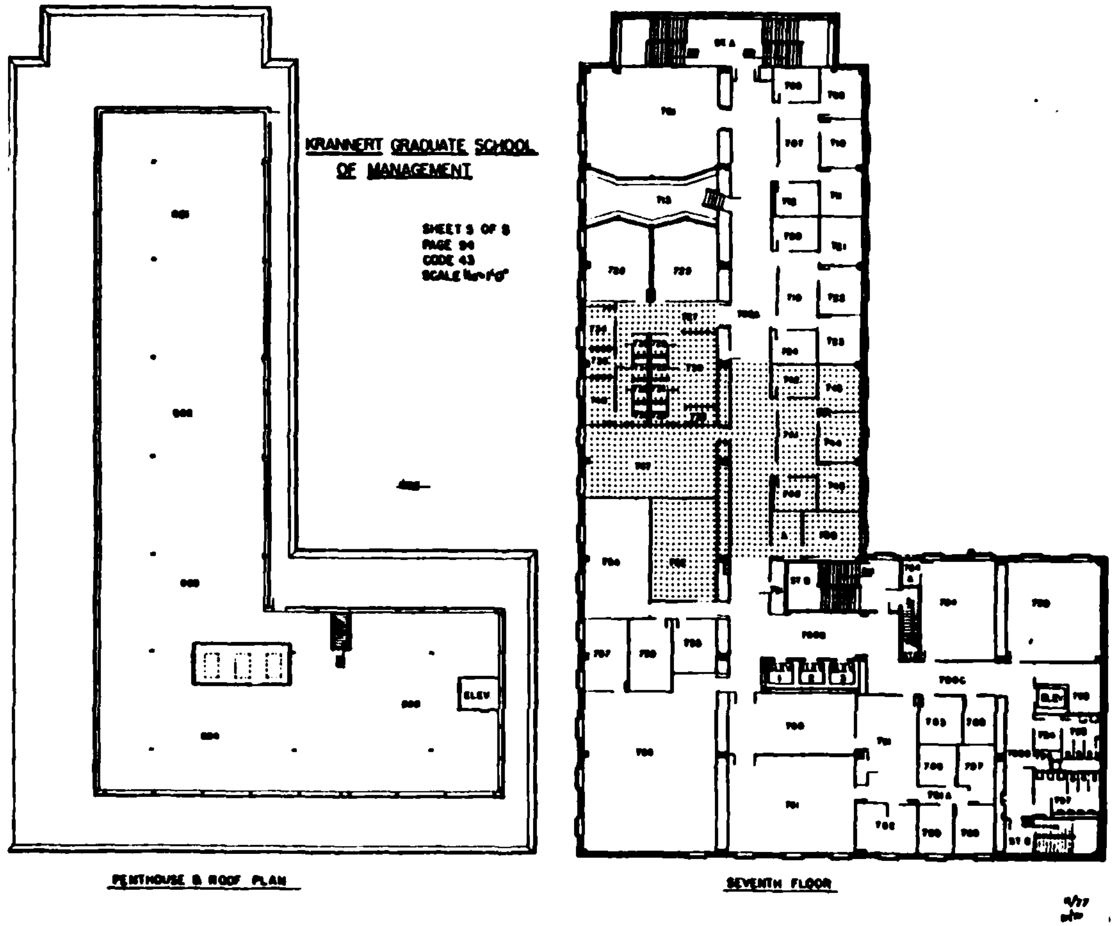


Figure A.4

Krannert Floor Plans: Seventh Floor And Penthouse

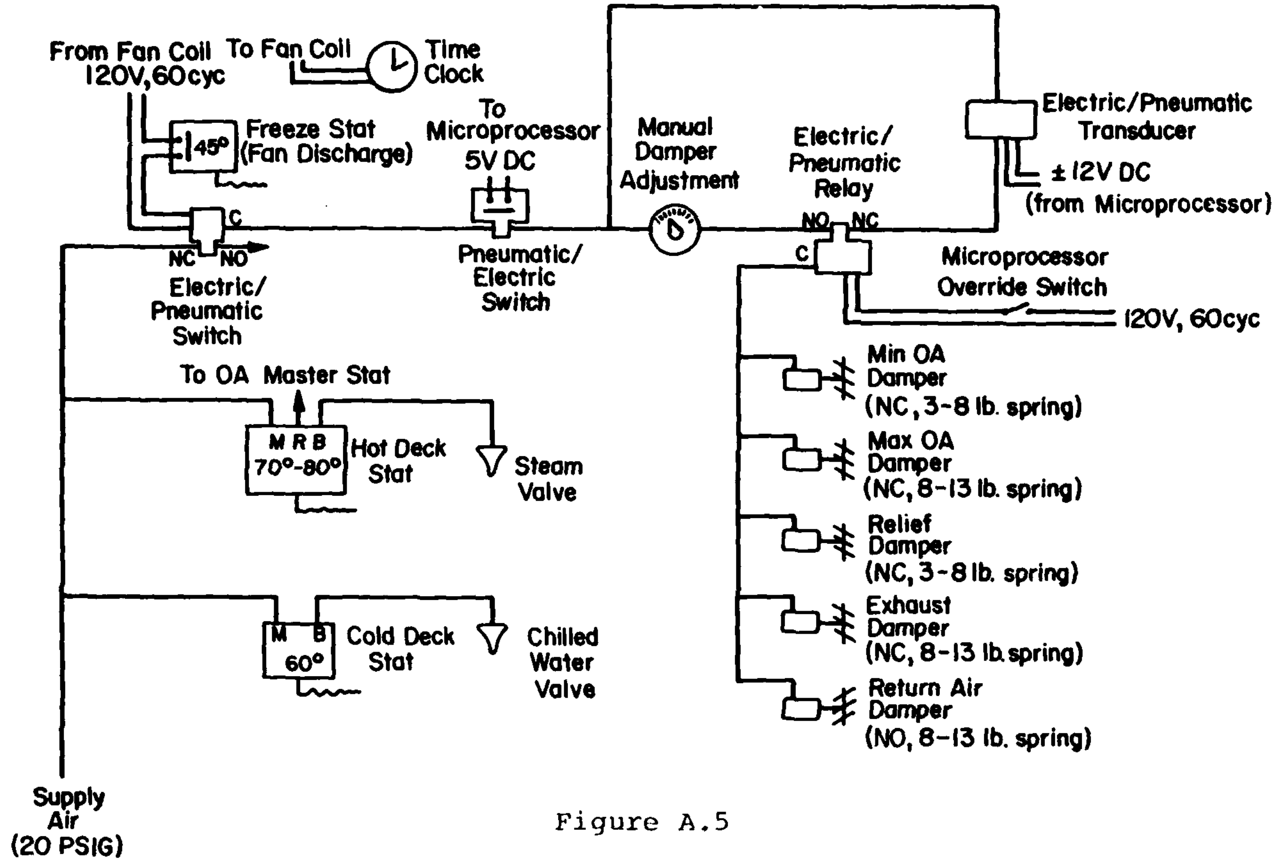


Figure A.5

Control Schematic For System ACP-7

Appendix B: Program PDECK

Appendix B: Program PDECK

```
      program main (input,output,tape5=input,tape6=output)
      common p
c
c  This program calculates the flows, costs, and psychrometic
c  data for the heating and cooling decks in HVAC system
c  ACP-7 in the Krannert Building using bin data for outside
c  air conditions.  It assumes the room load includes a
c  fixed latent load of 3 gr./lb dry air (times the supply
c  air flow), a fixed sensible load of 273,402 B/hr, and a
c  variable sensible load of 5,082 B/hr F times (ouside air
c  temp - inside air temp).  Subroutine Psyc is used to
c  perform the psychrometric calculations.  The value used
c  for the system cfm is based upon mixed air conditions.
c  The amount of outside air admitted is determined by an
c  algorithm which simulates microprocessor control.  This
c  algorithm compares the dry bulb temperatures of the
c  outside and return air, calculates the mixed air temp
c  which would result with vaious percentages of outdoor
c  air, and finds the percentage which would yield the
c  minimum operating cost.
c
c
c     abbreviations used in variable names:
c
c  oa = outside air
c  ma = mixed air
c  hd = hot deck
c  cd = cold deck
c  ra = room (or return) air
c  sa = supply air
c
c
c     variables used in air calculations
c
c  t__ = dry bulb temp (deg. f)
c  example:  toa = outside air dry bulb temp
c  tw__ = wet bulb temp (deg. f)
c  td__ = dew point temp (deg. f)
c  rh__ = relative humidity
c  w__ = specific humidity (grains/lbda)
c  h__ = enthalpy (btu/lbda)
c  v__ = specific volume (cu.ft./lbda)
c  pv__ = vapor pressure (in. hg.)
c
c
```

```
c      variables used in general calculations
c
c   cdc = cold deck cost, this bin ($/hr)
c   cdset = cold deck setpoint (deg. f)
c   cfmcd = required cfm through cold deck (cu.ft./min)
c   cfmhd = required cfm through hot deck (cu.ft./min)
c   cfmoa = cfm of outdoor air admitted (cu.ft./min)
c   cs = cost savings when using economizer, all bins totalled ($)
c   diff = difference between calculated room air humidity and
c           supply air humidity + room latent load (grains/lbda)
c   ecdct = total cold deck cost for all bins using economizer ($)
c   ehdct = total hot deck cost for all bins using economizer ($)
c   hdc = hot deck cost, this bin ($/hr)
c   hdset = hot deck setpoint (deg. f)
c   hrs = number of hours toa and twb occur this month (hr)
c   icount = counting variable to space outputs on pages
c   iflag = flag to show if economizer is on (=5) or off (=0)
c   it = number of iterations through this loop
c   month = month for which calculations are being made (alpha)
c   p = barometric pressure (in. hg.)
c   phd = percent of supply air going through hot deck
c   poa = percent outside air admitted
c   poamin = minimum percent outside air permitted
c   rload = room sensible load (btu/min)
c   scfm = supply air flow (cu.ft./min)
c   sp = room setpoint (deg. f)
c   tc = total cost of operating both decks, this bin ($)
c   tcdc = total cost to operate cold deck, this bin ($)
c   tcdcp = total cold deck cost, this bin, with previous % oa ($)
c   tcdct = total cold deck cost for all bins on minimum oa ($)
c   tcp = total cost for both decks, all bins, with previous % oa
c   tct = total cost for both decks, all bins, over entire month ($)
c   thdc = total hot deck cost, this bin ($)
c   thdcp = total hot deck cost, this bin, with previous % oa ($)
c   thdct = total hot deck cost for all bins using minimum oa ($)
c
c
c      variables used to simulate microprocessor calculations
c
c   cdcost = predicted cold deck cost on economizer ($/min *10-7)
c   cost = predicted total cost on economizer ($/min *10-7)
c   hdcost = predicted hot deck cost on economizer ($/min *10-7)
c   optpoa = optimum percent oa (yields lowest total cost)
c   optcost = total cost using optimum percent oa
c  treq = required supply air temp to meet room load (deg. f)
c
c
c             input data:
c
      p=29.92
      sp=75.0
      cdset=60.7
```

```
      hdset=80.0
      scfm=17355.0
      tct=0.0
      poamin=0.17
      tcp=0.0
      thdcp=0.0
      tcdcp=0.0
      thdct=0.0
      tcdct=0.0
      ehdct=0.0
      ecdct=0.0
      cs=0.0
      icount=0
      read(5,10) month
 10   format(a10)
      write(6,20) month
 20   format (1h1,' calculations for ',a10)
c
c  start output on a new page if icount = 2
c
 30   if(icount.lt.2) go to 35
      write(6,32)
 32   format(1h1)
      icount=0
 35   read(5,40) toa, twoa, hrs
 40   format (3f10.0)
      if(toa.ge.900.0) go to 160
      write(6,45)
 45   format(//,' ********************************************',
     c '********************',//)
      write(6,50) toa, twoa, hrs
 50   format(' bin data:',/,5x,'dry bulb=',f9.3,5x,'wet bulb=',
     c f9.3,5x,'hours duration=',f9.3,//,' on minimum oa:',//)
c
c    set iflag to 0 and percent oa to minimum
c
      poa=poamin
      iflag=0
c
c  calculate outdoor air conditions
c
      call psyc(1,toa,twoa,tdoa,rhoa,woa,hoa,voa,pvoa)
c
c   calculate mixed air conditions (assume return air temp = setpoint
c   and ra humidity = ma humidity + 3gr/lbda for the first iteration)
c
 55   it=1
      tra=sp
      wma=(woa*poa+3.0*(1.0-poa))/poa
      wra=wma+3.0
c
c   use mixing equations to deterine ma conditions for each iteration
```

```
c
  56  wma=poa*woa+(1.0-poa)*wra
      tma=poa*toa+(1.0-poa)*tra
      call psyc(3,tma,twma,tdma,rhma,wma,hma,vma,pvma)
c
c   calculate hot deck conditions
c
      thd=hdset
      if(tma.gt.thd)  thd=tma
      whd=wma
      call psyc(3,thd,twhd,tdhd,rhhd,whd,hhd,vhd,pvhd)
c
c   calculate cold deck conditions (assume condensation occurs,
c    then check assumption by comparing humidities)
c
      tcd=cdset
      if(tma.lt.tcd)  tcd=tma
      twcd=59.4
      if(tcd.le.twcd) twcd=tcd
      call psyc(1,tcd,twcd,tdcd,rhcd,wcd,hcd,vcd,pvcd)
      if (wma.gt.wcd) go to 59
      wcd=wma
      call psyc(3,tcd,twcd,tdcd,rhcd,wcd,hcd,vcd,pvcd)
  59  continue
c
c   calculate room load (per minute)
c
      rload=(273402.0-5082.0*(tra-toa))/60.0
c
c   calculate required supply air temp based upon dry air enthalpy
c   (if >thd or <tcd, assume max flow thru appropriate deck)
c
      tsa=tra-rload*vma/(scfm*0.24)
      if(tsa.lt.thd) go to 57
      phd=1.0
      go to 311
  57  if(tsa.gt.tcd) go to 58
      phd=0.0
      go to 311
  58  continue
c
c  calculate percent flow through hot deck and supply air humidity
c
      phd=(tsa-tcd)/(thd-tcd)
  311 wsa=phd*whd+(1.0-phd)*wcd
c
c   does room air humidity = supply air humidity + 3gr/lb?  if not,
c   adjust room air humidity and try again.
c
      diff=wra-(wsa+3.0)
      if(diff.gt.-0.5.and.diff.lt.0.5) go to 54
      wra=wsa+3.0
```

```
      it=it+1
      if(it.gt.20) print*,' return air iteration does not converge'
      if(it.gt.20) go to 54
      go to 56
c
c   calculate room air conditions
c
 54   wra=wsa+3.0
      call psyc(3,tra,twra,tdra,rhra,wra,hra,vra,pvra)
c
c   calculate required supply air conditions
c
      hsa=hra-rload*vma/scfm
c
c   if required supply air enthalpy > hot deck enthalpy, drop
c   room temp 0.1 degree and recalculate system air conditions
c
      if(hsa.le.hhd) go to 320
      tra=tra-0.1
      go to 56
320   continue
c
c   if required supply air enthalpy < cold deck enthalpy, raise
c   room temp 0.1 degree and recalculate system air conditions
c
      if(hsa.ge.hcd)  go to 330
      tra=tra+0.1
      go to 56
330   continue
c
c   calculate required cfm through hot and cold decks
c
      cfmhd=(hsa-hcd)*(scfm/vma)*vhd/(hhd-hcd)
      cfmcd=(scfm/vma-cfmhd/vhd)*vcd
c
c   calculate costs for each deck (per hour) and total cost per month
c
      hdc=((hhd-hma)*cfmhd/vhd)*(2.9*10.0**(-6.0))*(60.0)
      cdc=((hma-hcd)*cfmcd/vcd)*(6.58*10.0**(-6.0))*(60.0)
      if(hdc.lt.0.0) hdc=0.0
      if(cdc.lt.0.0) cdc=0.0
      thdc=hdc*hrs
      tcdc=cdc*hrs
      tc=thdc+tcdc
c
c   calculate a running total for costs in all bins this month
c
      if(iflag.gt.1) go to 65
      tct=tct+tc
      thdct=thdct+thdc
      tcdct=tcdct+tcdc
      ehdct=ehdct+thdc
```

```
      ecdct=ecdct+tcdc
      go to 66
   65 cs=cs+tcp-tc
      ehdct=ehdct+thdc-thdcp
      ecdct=ecdct+tcdc-tcdcp
   66 thdcp=thdc
      tcdcp=tcdc
      tcp=tc
c
c   print summary of results
c
      icount=icount+1
      write(6,60)
   60 format(12x,'outdoor air',4x,'room air',4x,'mixed air',4x,
     c 'hot deck',4x,'cold deck',/)
      write(6,70)toa,tra,tma,thd,tcd
   70 format(' temp',7x,f9.3,4x,f9.3,4x,f9.3,4x,f9.3,4x,f9.3)
      write(6,75) woa,wra,wma,whd,wcd
   75 format(' humidity',3x,f9.3,4x,f9.3,4x,f9.3,4x,f9.3,4x,f9.3)
      write(6,80) rhoa,rhra,rhma,rhhd,rhcd
   80 format(' percent rh',1x,f9.3,4x,f9.3,4x,f9.3,4x,f9.3,4x,f9.3)
      write(6,90) hoa,hra,hma,hhd,hcd
   90 format(' enthalpy',3x,f9.3,4x,f9.3,4x,f9.3,4x,f9.3,4x,f9.3)
      wwma=wma-0.01
      if(wcd.ge.wwma) go to 95
      write (6,91)
   91 format (//,' condensation occured')
   95 continue
      write(6,100) cfmhd,hdc
  100 format(//,' hot deck runs at ',f9.3,' cfm and costs', f9.4,
     c ' per hour')
      write(6,110) cfmcd,cdc
  110 format(' cold deck runs at ',f9.3,' cfm and costs',f9.4,
     c ' per hour')
      write(6,120) tc,thdc,tcdc
  120 format(' total monthly cost for both decks (this bin) is',f9.4,
     c //,'   (hot deck =',f9.4,'   cold deck =',f9.4,')',)
      if (iflag.gt.1) go to 30
c
c   calculate optimum oa admitted in economizer cycle
c
      iflag=5
      if (toa.gt.72.0) go to 166
      optcost=9999999999.9
      treq=(cfmhd*thd+cfmcd*tcd)/scfm
      optpoa=poa
  145 if(poa.gt.1.0) poa=1.0
      tma=poa*toa+(1.0-poa)*tra
      thd=hdset
      tcd=cdset
      if (tma.gt.thd) thd=tma
      if (tma.lt.tcd) tcd=tma
```

```
      cfmhd=scfm*(treq-tcd)/(thd-tcd)
      cfmcd=scfm-cfmhd
      hdcost=cfmhd*(thd-tma)*0.508
      cdcost=cfmcd*(tma-tcd)*1.19
      cost=hdcost+cdcost
      if (cost.lt.optcost) optpoa=poa
      if (cost.lt.optcost) optcost=cost
      if (poa.gt.0.999) go to 147
      poa=poa+0.01
      go to 145
  147 poa=optpoa
      if (poa.gt.poamin) go to 146
  166 write (6,130)
  130 format (//,' microprocessor would not increase outside air')
      go to 30
  146 cfmoa=scfm*poa
      write(6,150) cfmoa,poa
  150 format(//,' economizer operation selected',//,' with dampers',
     c ' set for ',f7.1,' cfm or ',f5.3,' percent oa:',//)
      go to 55
c
c  print total costs for entire month
c
  160 ect=ehdct+ecdct
      write(6,170) month,tct,thdct,tcdct,ect,ehdct,ecdct,cs
  170 format('1',///,' summary for the month of ',a10,///,
     c ' if running on min oa only the total cost for both decks is ',
     c f9.4,//,'   (hot deck = ',f9.4,'   cold deck = ',f9.4,')',///,
     c ' if used microprocessor control the total cost would be ',
     c f9.4,//,'   (hot deck = ',f9.4,' cold deck = ',f9.4,')',///,
     c ' total savings with microprocessor = ',f9.4)
      stop
      end
      subroutine psyc (key,t,tw,td,rn,w,h,v,pv)
c
c        psychrometric chart generator
c
c        list of variables
c        key specifies inputs
c        t   dry bulb temperature, deg f
c        tw  wet bulb temperature, deg f
c        td  dew point temperature, deg f
c        rh  relative humidity
c        w   humidity ratio, gr/lbda
c        h   enthalpy, btu/lbda
c        v   specific volume, cuft/lbda
c        pv  vapor pressure, in hg
c        p   barometric pressure, in hg
c
      common p
      go to (100,200,300,400,500,600,700),key
c
```

```
c         input--dry bulb and wet bulb
c
 100   if (tw.gt.t+0.01) print*,'error, wet bulb cannot be greater',
      1' than dry bulb--check inputs'
       pvw=gps(tw)
       hfw=ghf(tw)
       hg=ghg(t)
       w=(0.2403*(tw-t)+gw(pvw)*(ghg(tw)-hfw))/(hg-hfw)
       pv=gpv(w)
       rh=pv/gps(t)
       h=0.2403*t+w*hg
       go to 307
c
c         input--dry bulb and relative humidity
c
 200   pv=rh*gps(t)
       w=gw(pv)
       go to 304
c
c         input--dry bulb and humidity ratio
c
 300   w=w/7000.0
       pv=gpv(w)
       rh=pv/gps(t)
 304   hg=ghg(t)
       h=0.2403*t+w*hg
 306   tw=gwb(t,hg,w)
 307   td=tps(pv)
       if (t.lt.-60.0.or.t.gt.200.0) print *,'caution, dry bulb',
      1' outside range'
       if (td.lt.-60.0.or.td.gt.200.0) print *,'caution, dew point',
      1' outside range'
       v=0.7541*(t+459.67)/(p-pv)
       w=w*7000.0
       return
c
c         input--dry bulb and dew point
c
 400   pv=gps(td)
       rh=pv/gps(t)
       w=gw(pv)
       go to 304
c
c         input--dry bulb and enthalpy
c
 500   hg=ghg(t)
       w=(h-0.2403*t)/hg
       pv=gpv(w)
       rh=pv/gps(t)
       go to 306
c
c         input--enthalpy and relative humidity
```

```
c
 600  t=(h+700.0*rh/p)/(0.2403+20.0*rh/p)
      n=1
  10  pv=rh*gps(t)
      w=gw(pv)
      hg=ghg(t)
      y=h-0.2403*t-w*hg
      if (n.ge.50) go to 20
      if (abs(y).lt.0.0001) go to 30
      t=t+y/(37.5*w+0.2403)
      n=n+1
      go to 10
  20  print *,'iteration for dry bulb does not converge'
  30  go to 306
c
c       input--enthalpy and humidity ratio
c
 700  w=w/7000.0
      a=(0.44351+0.2403/w)/1.9194e-4
      b=(h/w-1061.19)/9.5971e-5
      t=a-sqrt(a*a-b)
      pv=gpv(w)
      rh=pv/gps(t)
      hg=ghg(t)
      go to 306
      end

      function ghf(t)
      if (t.lt.32.0) then
      ghf=-158.94+0.47123*t+4.923e-4*(t*t)
      else
      ghf=-31.924+0.99951*t
      endif
      if (t.gt.31.99.and.t.lt.32.01) print*,'gaseous water assumed',
     1 ' at dry bulb equal to 32 degrees (f)'
      return
      end

      function ghg(t)
      ghg=1061.19+0.44351*t-9.5971e-5*(t*t)
      return
      end

      function gps(t)
      ta=t+459.67
      if (t.lt.32.0) then
      gps=exp(20.807-11071.3/ta)
      else
      gps=exp(15.123-6766.3/ta-743166.9/(ta*ta))
      endif
      return
      end
```

```
      function tps(ps)
      if (ps.lt.0.0000001) go to 10
      if (ps.lt.0.180479) then
      tps=11071.3/(20.807-alog(ps))-459.67
      else
      a=15.123-alog(ps)
      b=3383.15/a
      c=743166.9/a
      tps=b+sqrt(b*b+c)-459.67
      endif
10    if (ps.lt.0.0000001) then
      tps=-100.0
      print*,'dew point undefined for zero relative humidity'
      endif
      return
      end

      function gwb(t,hg,w)
      common p
      tw=t
      n=1
10    psw=gps(tw)
      ww=gw(psw)
      y=w*hg-ww*ghg(tw)-(w-ww)*ghf(tw)+0.2403*(t-tw)
      if (n.ge.50) go to 20
      if (abs(y).lt.0.0001) go to 30
      tw=tw+y/(37.5*ww+w+0.2403)
      n=n+1
      go to 10
20    print *,'iteration for wet bulb does not converge'
30    gwb=tw
      if (tw.lt.-60.0.or.tw.gt.200.0) print*,'caution, wet bulb',
     1 ' outside range'
      if (tw.gt.t+0.01) print*,'error, wet bulb greater than',
     1 ' dry bulb'
      return
      end
      function gw(pv)
      common p
      gw=0.62202*pv/(p-pv)
      return
      end
      function gpv(w)
      common p
      gpv=p/(0.62202/w+1.0)
      return
      end
```

Appendix C: Program EDECK

Appendex C: Program EDECK

```
      program main (input,output,tape5=input,tape6=output)
      common p
c
c  This program calculates the flows, costs, and psychrometric
c  data for the heating and cooling decks in HVAC system
c  ACP-7 in the Krannert Building using bin data for outside
c  air conditions.  It assumes the room load includes a fixed
c  latent load of 3 grains/lb (times the supply air flow), a
c  fixed sensible load of 273,402 B/hr, and a variable sensible
c  load of 5,082 B/hr F times (outside air temp minus inside
c  air temp).  Subroutine Psyc is used to perform the psychro-
c  metric calculations.  The value used for the system cfm is
c  based upon mixed air conditions.
c  The program calculates the costs for all possible mixtures
c  of outside air and return air (in 1 percent increments) and
c  prints the costs for minimum outside air and optimum outside
c  air (if different).  In this manner it simulates the actions
c  of an ideal enthalpy economizer which could predict the
c  system's response and choose the optimum percent outdoor air.
c
c
c     abbreviations used in variable names:
c
c  oa = outside air
c  ma = mixed air
c  hd = hot deck
c  cd = cold deck
c  ra = room (or return) air
c  sa = supply air
c
c
c     variables used in air calculations
c
c  t__ = dry bulb temp (deg. f)  ex: toa = outside air dry bulb temp
c  tw__ = wet bulb temp (deg. f)
c  td__ = dew point temp (deg. f)
c  rh__ = relative humidity
c  w__ = specific humidity (grains/lbda)
c  h__ = enthalpy (btu/lbda)
c  v__ = specific volume (cu.ft./lbda)
c  pv__ = vapor pressure (in. hg.)
c
c
```

```
c     variables used in general calculations
c
c  cdc = cold deck cost, this bin ($/hr)
c  cdset = cold deck setpoint (deg. f)
c  cfmcd = required cfm through cold deck (cu.ft./min)
c  cfmhd = required cfm through hot deck (cu.ft./min)
c  cfmoa = cfm of outdoor air admitted (cu.ft./min)
c  cs = cost savings when using economizer, all bins totalled ($)
c  diff = difference between calculated room air humidity and
c         supply air humidity + room latent load (grains/lbda)
c  ecdct = total cold deck cost for all bins using economizer ($)
c  ehdct = total hot deck cost for all bins using economizer ($)
c  hdc = hot deck cost, this bin ($/hr)
c  hdset = hot deck setpoint (deg. f)
c  hrs = number of hours toa and twb occur this month (hr)
c  icount = counting variable to space outputs on pages
c  iflag = flag to show if economizer is on (=5) or off (=0)
c  it = number of iterations through this loop
c  month = month for which calculations are being made (alpha)
c  p = barometric pressure (in. hg.)
c  phd = percent of supply air going through hot deck
c  poa = percent outside air admitted
c  poamin = minimum percent outside air permitted
c  rload = room sensible load (btu/min)
c  scfm = supply air flow (cu.ft./min)
c  sp = room setpoint (deg. f)
c  tc = total cost of operating both decks, this bin ($)
c  tcdc = total cost to operate cold deck, this bin ($)
c  tcdcp = total cold deck cost, this bin, with previous % oa ($)
c  tcdct = total cold deck cost for all bins on minimum oa ($)
c  tcp = total cost for both decks, all bins, with previous % oa
c  tct = total cost for both decks, all bins, over entire month ($)
c  thdc = total hot deck cost, this bin ($)
c  thdcp = total hot deck cost, this bin, with previous % oa ($)
c  thdct = total hot deck cost for all bins using minimum oa ($)
c
c
c     variables used to simulate enthalpy controller calculations
c
c  optpoa = optimum percent oa (yields lowest total cost)
c  optcost = total cost using optimum percent oa
c
c
c            input data:
c
      p=29.92
      sp=75.0
      cdset=60.7
      hdset=80.0
      scfm=17355.0
      tct=0.0
      poamin=0.17
```

```
      tcp=0.0
      thdcp=0.0
      tcdcp=0.0
      thdct=0.0
      tcdct=0.0
      ehdct=0.0
      ecdct=0.0
      cs=0.0
      icount=0
      read(5,10) month
   10 format(a10)
      write(6,20) month
   20 format (1h1,' calculations for ',a10)
c
c  start output on a new page if icount = 2
c
   30 if(icount.lt.2) go to 35
      write(6,32)
   32 format(1h1)
      icount=0
   35 read(5,40) toa, twoa, hrs
   40 format (3f10.0)
      if(toa.ge.900.0) go to 160
      write(6,45)
   45 format(//,' ********************************************',
     c '********************',//)
      write(6,50) toa, twoa, hrs
   50 format(' bin data:',/,5x,'dry bulb=',f9.3,5x,'wet bulb=',
     c f9.3,5x,'hours duration=',f9.3,//,' on minimum oa:',//)
c
c    set iflag to 0 and percent oa to minimum
c
      poa=poamin
      iflag=0
c
c  calculate outdoor air conditions
c
      call psyc(1,toa,twoa,tdoa,rhoa,woa,hoa,voa,pvoa)
c
c   calculate mixed air conditions (assume return air temp = setpoint
c   and ra humidity = ma humidity + 3gr/lbda for the first iteration)
c
   55 it=1
      tra=sp
      wma=(woa*poa+3.0*(1.0-poa))/poa
      wra=wma+3.0
c
c   use mixing equations to deterine ma conditions for each iteration
c
   56 wma=poa*woa+(1.0-poa)*wra
      tma=poa*toa+(1.0-poa)*tra
      call psyc(3,tma,twma,tdma,rhma,wma,hma,vma,pvma)
```

```
c
c   calculate hot deck conditions
c
      thd=hdset
      if(tma.gt.thd)  thd=tma
      whd=wma
      call psyc(3,thd,twhd,tdhd,rhhd,whd,hhd,vhd,pvhd)
c
c   calculate cold deck conditions (assume condensation occurs,
c    then check assumption by comparing humidities)
c
      tcd=cdset
      if(tma.lt.tcd)  tcd=tma
      twcd=59.4
      if(tcd.le.twcd) twcd=tcd
      call psyc(1,tcd,twcd,tdcd,rhcd,wcd,hcd,vcd,pvcd)
      if (wma.gt.wcd) go to 59
      wcd=wma
      call psyc(3,tcd,twcd,tdcd,rhcd,wcd,hcd,vcd,pvcd)
  59  continue
c
c   calculate room load (per minute)
c
      rload=(273402.0-5082.0*(tra-toa))/60.0
c
c   calculate required supply air temp based upon dry air enthalpy
c   (if >thd or <tcd, assume max flow thru appropriate deck)
c
      tsa=tra-rload*vma/(scfm*0.24)
      if(tsa.lt.thd) go to 57
      phd=1.0
      go to 311
  57  if(tsa.gt.tcd) go to 58
      phd=0.0
      go to 311
  58  continue
c
c  calculate percent flow through hot deck and supply air humidity
c
      phd=(tsa-tcd)/(thd-tcd)
  311 wsa=phd*whd+(1.0-phd)*wcd
c
c   does room air humidity = supply air humidity + 3gr/lb?  if not,
c   adjust room air humidity and try again.
c
      diff=wra-(wsa+3.0)
      if(diff.gt.-0.5.and.diff.lt.0.5) go to 54
      wra=wsa+3.0
      it=it+1
      if(it.gt.20) print*,' return air iteration does not converge'
      if(it.gt.20) go to 54
      go to 56
```

```
c
c   calculate room air conditions
c
  54  wra=wsa+3.0
      call psyc(3,tra,twra,tdra,rhra,wra,hra,vra,pvra)
c
c   calculate required supply air conditions
c
      hsa=hra-rload*vma/scfm
c
c   if required supply air enthalpy > hot deck enthalpy, drop
c   room temp 0.1 degree and recalculate system air conditions
c
      if(hsa.le.hhd) go to 320
      tra=tra-0.1
      go to 56
 320  continue
c
c   if required supply air enthalpy < cold deck enthalpy, raise
c   room temp 0.1 degree and recalculate system air conditions
c
      if(hsa.ge.hcd)  go to 330
      tra=tra+0.1
      go to 56
 330  continue
c
c   calculate required cfm through hot and cold decks
c
      cfmhd=(hsa-hcd)*(scfm/vma)*vhd/(hhd-hcd)
      cfmcd=(scfm/vma-cfmhd/vhd)*vcd
c
c   calculate costs for each deck (per hour) and total cost per month
c
      hdc=((hhd-hma)*cfmhd/vhd)*(2.9*10.0**(-6.0))*(60.0)
      cdc=((hma-hcd)*cfmcd/vcd)*(6.58*10.0**(-6.0))*(60.0)
      if(hdc.lt.0.0) hdc=0.0
      if(cdc.lt.0.0) cdc=0.0
      thdc=hdc*hrs
      tcdc=cdc*hrs
      tc=thdc+tcdc
      if(iflag.gt.4) go to 145
c
c   calculate a running total for costs in all bins this month
c
      if(iflag.gt.1) go to 65
      tct=tct+tc
      thdct=thdct+thdc
      tcdct=tcdct+tcdc
      ehdct=ehdct+thdc
      ecdct=ecdct+tcdc
      go to 66
  65  cs=cs+tcp-tc
```

```
      ehdct=ehdct+thdc-thdcp
      ecdct=ecdct+tcdc-tcdcp
 66   thdcp=thdc
      tcdcp=tcdc
      tcp=tc
c
c   print summary of results
c
      icount=icount+1
      write(6,60)
 60   format(12x,'outdoor air',4x,'room air',4x,'mixed air',4x,
     c 'hot deck',4x,'cold deck',/)
      write(6,70)toa,tra,tma,thd,tcd
 70   format(' temp',7x,f9.3,4x,f9.3,4x,f9.3,4x,f9.3,4x,f9.3)
      write(6,75) woa,wra,wma,whd,wcd
 75   format(' humidity',3x,f9.3,4x,f9.3,4x,f9.3,4x,f9.3,4x,f9.3)
      write(6,80) rhoa,rhra,rhma,rhhd,rhcd
 80   format(' percent rh',1x,f9.3,4x,f9.3,4x,f9.3,4x,f9.3,4x,f9.3)
      write(6,90) hoa,hra,hma,hhd,hcd
 90   format(' enthalpy',3x,f9.3,4x,f9.3,4x,f9.3,4x,f9.3,4x,f9.3)
      wwma=wma-0.01
      if(wcd.ge.wwma) go to 95
      write (6,91)
 91   format (//,' condensation occured')
 95   continue
      write(6,100) cfmhd,hdc
 100  format(//,' hot deck runs at ',f9.3,' cfm and costs', f9.4,
     c ' per hour')
      write(6,110) cfmcd,cdc
 110  format(' cold deck runs at ',f9.3,' cfm and costs',f9.4,
     c ' per hour')
      write(6,120) tc,thdc,tcdc
 120  format(' total monthly cost for both decks (this bin) is',f9.4,
     c //,'   (hot deck =',f9.4,'   cold deck =',f9.4,')',)
      if (iflag.gt.1) go to 30
c
c   calculate optimum oa admitted by enthalpy controller
c
      iflag=5
      optpoa=poa
      optcost=tc
      if (hoa.gt.hra) go to 147
 145  if (tc.lt.optcost) optpoa=poa
      if (tc.lt.optcost) optcost=tc
      if (poa.gt.0.999) go to 147
      if (tma.lt.59.0) go to 147
      poa=poa+0.01
      if (poa.gt.1.0) poa=1.0
      go to 55
 147  poa=optpoa
      if (poa.gt.poamin) go to 146
      write (6,130)
```

```
  130 format (//,' enthalpy controller would not increase outside air')
      go to 30
  146 cfmoa=scfm*poa
      write(6,150) cfmoa,poa
  150 format(//,' economizer operation selected',//,' with dampers',
     c ' set for ',f7.1,' cfm or ',f5.3,' percent oa:',//)
      iflag=2
      go to 55
c
c  print total costs for entire month
c
  160 ect=ehdct+ecdct
      write(6,170) month,tct,thdct,tcdct,ect,ehdct,ecdct,cs
  170 format('1',///,' summary for the month of ',a10,///,
     c ' if running on min oa only the total cost for both decks is ',
     c f9.4,//,'    (hot deck = ',f9.4,'    cold deck = ',f9.4,')',///,
     c ' if used enthalpy control the total cost would be ',f9.4,
     c //,'    (hot deck = ',f9.4,' cold deck = ',f9.4,')',///,
     c ' total savings with microprocessor = ',f9.4)
      stop
      end
      subroutine psyc (key,t,tw,td,rh,w,h,v,pv)
c
<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>


    NOTE:  The remainder of subroutine psyc has
    been omitted from this thesis, as it is identical
    to the same subroutine in program Pdeck


<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
```

Appendix D: Program EMASTER

Appendix D: Program EMASTER

```
*************************************************************
*
*
*  PROGRAM DESCRIPTION:  This program reads temperature and
*  flow data from an air conditioning system and controls
*  the outside air dampers to minimize the cost of running
*  the hot and cold decks.  An interrupt 3 service routine
*  maintains a 24 hour clock and spaces control actions at
*  regular intervals.  The feedback/control loop is executed
*  every 30 seconds, and every 15 minutes data is read and
*  a new mixed air temperature setpoint is calculated.  A
*  second microprocessor (micro2) is used to read the flow
*  sensors and to log the collected data.  Communications
*  between the two microprocessors is done via the serial
*  port using standard XOP's at 300 baud.
*
*  A "pulse width modulated" signal is used to control the
*  damper position, i.e. at regular intervals the transducer
*  controlling the dampers is moved by a signal pulse and
*  the wider this pulse is the more the transducer moves.
*  A PID control algorithm is used to calculate the pulse
*  width (i.e. time on) and this number is used to time a
*  delay loop.  The appropriate "increase" or "decrease"
*  signal is sent to the transducer, the delay loop is
*  implemented, and then the signal is switched off.
*
*  Subroutine READ is employed to read data from the A/D
*  convertors, subroutine OUTAIR computes the optimum
*  mixed air setpoint, and subroutine CONTROL is used
*  to calculate the PID output.  Subroutine HEXDEC
*  performs  hex to decimal conversions, and
*  subroutines ADD and MULT perform overflow-protected
*  addition and multiplication.
*
*
*  register useage
*
*  main workspace  (at location mainwp)
*
*   r0       interrupt linking vector/misc
*   r1       timer value for 9901 clock/misc.
*   r2-r3    output pulse to transducer
*   r4       data XOP'd to and from micro2
*   r5-r7    Data storage and transmission
*   r8       previous clock time
```

```
*    r9        multiple uses during calculations
*    r10       number of delays to equal 15 minutes
*    r11       not used
*    r12       CRU base address
*    r13       number of 15 min delays between data logging
*    r14       number of interrupts between control actions
*    r15       misc
*
*   interrupt 3 workspace described with service routine
*
*   subroutine "hexdec" workspace described with subroutine
*
*   subroutine "outair" workspace described with subroutine
*
*   subroutine "control" workspace described with subroutine
*
*   subroutine "read" workspace described with subroutine
*
*   subroutine "add" workspace described with subroutine
*
*   subroutine "mult" workspace described with subroutine
*
*
*   interrupt useage:  interrupt 3 timer
*
*
*   i/o useage:  off-board A/D convertor
*                serial port communications with micro2
*
*
*   variable useage
*
*     ans      answer from subroutine "add" or "mult"
*     cdcost   cold deck cost ($/min) in current loop
*     cdpric   cold deck price ($/cfm-deg F x 10**-7)
*     cdsp     cold deck setpoint (deg F)
*     cdt      cold deck temp (deg F)
*     cdv      cold deck flow (100 cfm)
*     clock    current time (24 hr clock, hrs & min, decimal)
*     conout   control signal to be output
*     cost     minimum total cost ($/min) of hot & cold decks
*     deriv    derivitive gain for PID algorithm
*     diff     absolute value of RAT-OAT
*     hdcost   hot deck cost ($/min) in current loop
*     hdpric   hot deck price ($/cfm-deg F x 10**-7)
*     hdt      hot deck temp (deg F)
*     hdv      hot deck flow (100 cfm)
*     int      integral gain for PID algorithm
*     mat      mixed air temp (deg F)
*     maxpoa   maximum poa which dampers can admit
*     maxxoa   maximum poa allowed at present time
*     minpoa   minimum percent outside air allowable
```

```
*    num1     input value for subroutine "add" or "mult"
*    num2     input value for subroutine "add" or "mult"
*    oat      outside air temp (deg F)
*    optmat   optimum mixed air temp (deg F), using optpoa
*    optpoa   optimum percent outside air (calculated)
*    poa      percent outside air used in current calculation
*    preerr   previous error value
*    preint   previous integral sum
*    preset   previous setpoint value
*    prop     proportional gain for PID algorithm
*    rat      return air temp (deg F)
*    span0    span adjustment for sensor #0 (oat)
*    span1    span adjustment for sensor #1 (rat)
*    span2    span adjustment for sensor #2 (mat)
*    span3    span adjustment for sensor #3 (cdt)
*    span4    span adjustment for sensor #4 (hdt)
*
*  significant addresses (excludes local loop addresses)
*
*   add       linking vector for subroutine "add"
*   add1      first line of subroutine "add"
*   addwp     workspace for "add", "mult", and "read"
*   cont      linking vector for subroutine "control"
*   cont1     first line of subroutine "control"
*   enter     starting address for main control loop
*   hex1      first line of subroutine "hexdec"
*   hexdec    linking vector for subroutine "hexdec"
*   int3      first line of interrupt 3
*   int3wp    first line of interrupt 3 workspace
*   mainwp    first line of main workspace
*   mult      linking vector for subroutine "mult"
*   mult1     first line of subroutine "mult"
*   out1      first line of subroutine "outair"
*   outair    linking vector for subroutine "outair"
*   outwp1    first workspace for subroutine "outair"
*   outwp2    second workspace for subroutine "outair"
*   read      linking vector for subroutine "read"
*   read1     first line of subroutine "read"
*   start     first line of main program
*
*
*
*
      idt 'emaster'      output program id after download
      titl 'economizer control program (master)'
      option xref,symt   output cross ref. and symbol table
*
*
**********************************************************
*
*          main program
*
```

```
*   initialization routine
*
         aorg >0146        set starting address
start    lwpi mainwp       set main workspace pointer
*
*
*   Switch micro output from on-board display to serial port
*
*
         clr @>0036        clear keyboard display flag
         li r12,>0800      load r12 with address of 9902
         sbo 31            reset TMS 9902 UART
         li r0,>6a00       load r0 with control reg. code
         ldcr r0,8         initialize TMS 9902 control reg.
         sbz 13            do not int interval reg.
         li r1,>0341       set up for 300 baud
         ldcr r1,12        set up the TMS 9902
*
*
*   set up interrupt 3 linking vector
*
*
         li r0,int3wp      load address of int3 workspace
         mov r0,@>0004      into proper location
         li r0,int3        load address of first line of int3
         mov r0,@>0006      into proper location
*
*
*   load 9901 clock and enable interrupt 3
*
*
         li r12,>0000      load r12 with address of 9901 clock
         li r1,>fa11      load r1 for 0.5 sec. delay
*                         (bits 1-14 = >7d08 = delay counter
*                          and bit 15 = 1 to enable clock)
         ldcr r1,15        load & enable 9901 clock
         sbz 0             leave clock mode, enable inter. mode
         sbo 3             enable int3 (level 1 at micro)
         limi 1            enable interrupts 0-1 at micro
*
*
         li r1,120         load r1 for 120 .5 sec delays (1 min)
         mov r1,@int3wp+2  mov r1 to interrupt 3 r1
*
*   Intitate communication with micro2
*
        li r4,>4f4f       load r4 with ASCII "O"
        xop r4,12         send "on" signal to micro2
ready   xop r5,13         wait for "on" signal from micro2
        cb r5,r4          was signal ASCII "O"?
        jeq minoa         if so, jump to "minoa"
        li r5,>5858       if not, load r5 with ASCII "X"
```

```
        xop r5,12         send error signal to micro2
        jmp ready         try again
*
*
*  run system at minimum outside air on start-up
*   (allows temps and flows to stabilize)
*
*
minoa   mov @minpoa,@maxxoa   limit optimization loop to
*                          minpoa on first pass
         li r10,1          load delay counter to calculate
*                           OPTMAT on first pass
         li r13,3          load r13 to delay 30 min. before
*                           first data logging
         li r14,59         load r14 to take control actions
*                           every 30 seconds
        idle              wait for interrupt 3
*
*
*  Is air conditioning system on?
*
*
enter   li r12,>0020       load r12 for 9901 I/O ports
        tb 4               test bit 4 (high if A/C on)
        jne minoa          if A/C off, repeat loop
*
*
*  Transmit data to Micro2 every 1/2 hr. for logging
*    (transmit CLOCK,OAT,RAT,MAT,CDT,HDT,CDV,HDV,OPTMAT
*    and OPTPOA in sequence)
*
*
        dec r10             decrement 15 min. counter
        jne nolog           if not zero, jump
        li r10,1770         reset r10 for 15 minutes
        dec r13             decrement 30 min counter
        jne noxmit          if not zero, jump
        li r13,2            reset r13 for two 15 min cycles
w1      li r4,>5757         load r4 with ASCII "w"
        xop r4,12           send "write" symbol to micro2
        xop r4,13           wait for "ready" signal from micro2
        ci r4,>5200         compare r4 with ASCII "R"
        jne w1              if not equal, try again
*
*
*
        li r6,clock         load r6 with address of "clock"
        xop *r6+,10         xmit "clock" to micro2
        li r5,>0d0d         load r5 with ASCII carriage return
*
        xop r5,12           xmit "carriage return" to micro2
        li r1,9             load r1 for 9 additional data xmits
```

```
*
*
*
xmit    mov *r6+,r4       load data for hex/dec conversion
        blwp @hexdec      call hexdec
        xop r4,10         xmit decimal number to micro2
        xop r5,12         xmit "carriage return" to micro2
        dec r1            decrement transmission counter
        jne xmit          if not zero, repeat loop
*
*
*  Clear echo from xop9 in Micro2 and check for completion
*
*
        xop r4,13          read & clear echo from micro2
        li r4,>4700        load r4 with ASCII "G"
        xop r4,12          send "got it?" inquiry to micro2
        xop r4,13          wait for "logged" reply
        ci r4,>4c00        compare r4 with ASCIII "L"
        jne w1             if not equal, repeat transmission
*
*
*  read data and compute new setpoint every 15 minutes
*   (will read OAT, RAT, MAT, CDT, & HDT in order)
*
*
noxmit li r0,5            load r0 for 5 temp readings
        li r1,OAT         load r1 w/ addr. of 1'st temp
        li r15,0000       load r15 for A/D channel 0
ndat    mov r15,r9        copy A/D channel # in r9
        blwp @read        read temp
        mov r9,*r1+       store temp
        ai r15,>0100      set r15 for next A/D channel
        dec r0            decrement counter
        jne ndat          if not zero, read next temp
*
*
read2   li r4,>5252        load r4 with ASCII "R"
        xop r4,12          send "read" symbol to micro2
        xop r4,13          wait for "transmitting" signal
        ci r4,>5400        was signal ASCII "T"?
        jne read2          if not, repeat request
        xop r4,9           read hot deck flow from micro2
        data err           jump to err if have input error
        data err           jump to err if have input error
        mov r4,@hdv        store in "hdv"
        xop r4,9           read cold deck flow from micro2
        data err           jump to err if have input error
        data err           jump to err if have input error
        mov r4,@cdv        store in "cdv"
        xop r4,13          wait for "got it?" inquiry
        ci r4,>4700        compare r4 with ASCII "G"
```

```
       jne read2          if not equal, repeat
       li r4,>4c00        load r4 with ASCII "L"
       xop r4,12          send "logged" signal to micro2
       jmp sp             jump to "sp"
*
*
err    li r1,2            load R1 for two cycles
err1   li r5,>5800        load r5 with ASCII "X"
       xop r5,12          send "error" message to micro2
       xop r4,13          wait for data/reply from micro2
       dec r1             decrement counter
       jne err1           if not zero, repeat
       jmp read2          repeat read cycle
*
*
sp     blwp @outair       call "outair" to calculate new
*                          mixed air setpoint
       mov @maxpoa,@maxxoa  allow operation up to MAXPOA
*                                during remaining loops
*
*
*  Calculate control output and output control signal
*   every 30 seconds
*
*
nolog  dec r14            decrement interrupt counter
       jne noout          if not zero, jump to "no output"
       blwp @cont         call "control" for new output
       mov @conout,r2     store "control" output in r2
       ci r2,0000         compare output to zero
       jgt raise          if +, raise transducer output
       jlt drop           if -, drop transducer output
       clr r3             if 0, don't change output and
       jmp output         jump to "output"
raise  li r3,>0100        set r3 for increased output
       jmp output         jump to "output"
drop   li r3,>0200        set r3 for decreased output
output abs r2             take absolute value of "conout"
       idle               wait for interrupt 3
       li r12,>0020       set r12 for E/P transducer
       ldcr r3,2          output "raise" or "drop" signal
       ai r2,1            add 1 to r2 (so don't decrement 0)
dlay   dec r2             decrement r2 (delay counter)
       jne dlay           if r2 not 0, repeat delay loop
       clr r3             prepare to turn off transducer
       ldcr r3,2          turn off transducer motor
       li r14,59          reset interrupt counter
*
*
noout  idle                wait for interrupt 3
       b @enter            repeat loop
*
```

```
************************************************************************
*
************************************************************************
*
*
* subroutine outair
*
*  This subroutine fetches temperature and flow readings
*  from the main program and calculates the optimum
*  percent outside air to be admitted to the system based
*  upon the combined hot and cold deck costs.  Based upon
*  this optimum percent oa the subroutine calculates the
*  optimum mixed air temp which will serve as the setpoint
*  for the control routine.  The optimum mixed air temp is
*  stored in location "OPTMAT", and the optimum percent
*  outside air is stored in location "OPTPOA".  In addition
*  to these variables, storage locations "POA" and "COST"
*  must be provided in the main program.  The subroutine
*  requires data be supplied to it from the following
*  locations:
*
*       oat     outside air temp
*       rat     return air temp
*       cdt     cold deck temp
*       hdt     hot deck temp
*       cdv     cold deck flow
*       hdv     hot deck flow
*       minpoa  minimum percent oa allowable
*       maxpoa  maximum percent oa allowable
*       cdpric  cold deck price (cost/btu)
*       hdpric  hot deck price (cost/btu)
*
*  The subroutine requires the use of subroutines "ADD"
*  and "MULT" to prevent overflow when signed numbers
*  are added and multiplied.
*
*
* REGISTER USEAGE  This subroutine uses two workspaces
*
*  Register 1:  (at location "outwp1")
*
*  r0      outside air temp
*  r1      return air temp
*  r2      mixed air temp (calculated)
*  r3      cold deck temp
*  r4      hot deck temp
*  r5      cold deck flow
*  r6      hot deck flow
*  r7      total flow
*  r8      required supply air temp
*  r9      loop counter
*  r10     multiple uses during calculations
```

```
*   r11     multiple uses during calculations
*   r12     multiple uses during calculations
*   r13-15  return context switch data
*
*   Register 2:  (at location "outwp2")
*
*   r0      cold deck price
*   r1      hot deck price
*   r2      mixed air temp (calculated)
*   r3      cold deck temp
*   r4      hot deck temp
*   r5      required cold deck flow
*   r6      required hot deck flow
*   r7      total flow
*   r8      required supply air temp
*   r9      multiple uses during calculations
*   r10     total cost, both decks
*   r11-r15 multiple uses during calculations
*
*
*   Load Data Into Workspaces
*
out1    mov @oat,r0          move OA temp to r0
        mov @rat,r1          move RA temp to r1
        mov @cdt,r3          move cold deck temp to r3
        mov @hdt,r4          move hot deck temp to r4
        mov @cdv,r5          move cold deck flow to r5
        mov @hdv,r6          move hot deck flow to r6
*
*      Calculate supply air temperature required to maintain
*        room air conditions
*
        mov r5,r7            copy cold deck flow in r7
        a r6,r7              add total flow (both decks) r7
        mov r4,r10           copy hot deck temp in r10
        mpy r6,r10           multiply hot deck flow * temp
        div r7,r10           divide hd product by total flow
        mov r10,r8           copy quotient in r8
        mov r11,r12          copy remainder in r12
        mov r3,r10           copy cold deck temp in r10
        mpy r5,r10           multiply cold deck flow * temp
        div r7,r10           divide CD product by total flow
        a r10,r8             add quotient to r8 (=req'd temp)
        a r11,r12            add remainders in r12
        mov r7,r11           copy total flow in r11
        srl r11,1            divide total flow by 2
        c r12,r11            compare remainder to 1/2 divisor
        jle nornd            if remainder < 1/2 div..., jump
        ai r8,1              else round r8 up 1
*
*      Initialize data for optimization loop
*
```

```
nornd   mov @minpoa,@poa     set percent OA to minimum
        li r10,>7fff         load r10 with max possible + cost
        mov r10,@cost        set "cost" to max possible
        mov @maxxoa,r9       load loop counter for max POA
        mov @minpoa,r10      load r10 with min percent OA
        s r10,r9             subtract minpoa from loop counter
        ai r9,1              add 1 (so don't decrement 0)
*
*       Calculate mixed air temp with current percent oa
*
rpt     mov @poa,r10         load r10 with percent oa
        mov r0,r11           copy OA temp in r11
        abs r11              take abs. value OA temp
        c r0,r11             compare OAT with abs(OAT)
        jeq oapos            if equal, jump to OAPOS
        mpy r10,r11           else mult abs(OAT)*POA
        li r10,100           load r10 with 100 percent
        div r10,r11          r11 = OAT*POA/100 percent
        neg r11              negate r11 (since OAT -)
        jmp thru3            jump to thru3
oapos   mpy r10,r11          multiply OAT*POA
        li r10,100           load r10 with 100 percent
        div r10,r11          r11 = OAT*POA/100 percent
thru3   mov r11,r2           store quotient in r2
        mov @poa,r11         load r11 with percent OA
        s r11,r10            100 - percent OA = percent RA
        mpy r1,r10           percent RA * RAT in r10 and r11
        li r12,100           load r12 with 100 percent
        div r12,r10          r10 = RAT*Percent RA/100 percent
        a r10,r2             (POA * OAT + PRA * RAT)/100 = MAT
*
*   FREEZE PROTECTION:  If the mixed air temp drops below 38
*   degrees F, exit optimization routine and use optimum
*   mixed air temp calculated prior to this point.
*
        ci r2,380            compare MAT to 38 degrees F
        jlt freeze           if MAT < 380, exit routine
*
*       Will new mixed air temp affect deck temps?
*
        c r2,r4              compare hot deck & mixed air temps
        jlt hdok             if MAT < HDT, jump
        mov r2,r4             else set HDT = MAT
hdok    mov @cdsp,r3         reset r3 to cold deck setpoint
        c r3,r2              compare cold deck & mixed air temps
        jlt cdok             if CDT < MAT, jump
        mov r2,r3             else set CDT = MAT
*
*       Shift to second workspace
*
cdok    lwpi outwp2          shift to second workspace
        mov @cdpric,r0       move cold deck price to r0
```

```
        mov @hdpric,r1       move hot deck price to r1
        mov @outwp1+4,r2     copy MA temp into new R2
        mov @outwp1+6,r3     copy CD temp into new R3
        mov @outwp1+8,r4     copy HD temp into new R4
        mov @outwp1+14,r7    copy total flow into new R7
        mov @outwp1+16,r8    copy req'd SA temp into new R8
*
*       Calculate required flows to maintain supply temp
*
        mov r8,r9              copy req'd temp in r9
        s r3,r9                subtract cold deck temp
        mpy r7,r9              mult temp diff by total flow
        mov r3,r11             copy cold deck temp in r11
        mov r4,r13             copy hot deck temp in r13
        s r11,r13              subtract CD temp from HD temp
        div r13,r9             HDFLOW=FLOW*(TREQ-CDT)/(HDT-CDT)
        srl r13,1              divide r13 by two
        c r10,r13              compare remainder to 1/2 divisor
        jle nornd2             if remainder < 1/2 divisor, jump
        ai r9,1                else round quotient up by 1
nornd2  mov r9,r6              copy HD flow in r6
        mov r7,r5              copy total flow in r5
        s r6,r5                total flow - HD flow = CD flow
*
*       Calculate cost of running hot and cold decks
*
        mov r4,r9              copy hot deck temp in r9
        s r2,r9                subtract MA from HD temp
        mpy r6,r9              mult temp diff by HD flow
        li r15,10              load r15 with ten
        div r15,r9             divide product by 2
        mpy r1,r9              mult product by HD price
        li r15,100             load r15 with 100
        div r15,r9             r9=cost/100 (prevent overflow)
        mov r9,@hdcost         store cost in "hdcost"
        mov r2,r11             copy MA temp in r11
        s r3,r11               subtract CD from MA temp
        mpy r5,r11             mult temp diff by CD flow
        li r15,10              load r15 with ten
        div r15,r11            divide product by two
        mpy r0,r11             mult product by CD price
        li r15,100             load r15 with 100
        div r15,r11            r11=cost/100 (prevent overflow)
        mov r11,@cdcost        store cost in "cdcost"
        mov r11,@num1          move CD cost to "num1"
        mov r9,@num2           move HD cost to "num2"
        blwp @add              CD cost + HD cost = total cost
        mov @ans,r10           store total cost in r10
        c r10,@cost            compart total cost to low cost
        jgt notopt             if not lower, jump
        mov r10,@cost           else, set new lowest cost and
        mov r2,@optmat          new optimum Mixed Air temp and
```

```
       mov @poa,@optpoa       new optimum percent outside air
*
*      Shift back to original workspace
*
notopt lwpi outwp1            shift to workspace #1
*
*      If outside air temp > RAT - 3, exit loop on first
*      pass.  (will lock optpoa on minpoa)
*
       mov r1,r10             Store RAT in r10
       mov r0,r11             copy OAT in r11
       s r11,r10              r10 = RAT - OAT
       abs r10                take absolute value of r10
       mov r10,@diff          store difference
       ci r10,30              is difference > 3 Deg F?
       jlt freeze             if so, exit loop
       inc @poa               increment percent outside air
       dec r9                 decrement loop counter
       jne rpt2               if not zero, repeat loop
freeze rtwp                   return
rpt2   b @rpt                 repeat loop (too big for jne)
*
*
******************************************************************
*
************************************************************
*
*   Subroutine Control
*
*   Program Description:  This subroutine computes the error
*   between the actual mixed air temperature measured by the
*   sensor and the optimum mixed air temperature calculated
*   by subroutine "outair" and calculates a PID control
*   signal based upon this error.  The PID algorithm used
*   requires a constant sample rate, as this rate is
*   incorporated into the integral and derivitive constants.
*   Subroutines "ADD" and "MULT" are used to prevent overflow
*   when adding or multiplying, if the answer computed by
*   either subroutine exceeds the maximum positive or negative
*   number which can be written in 16 bits, the subroutine
*   assigns the maximum possible value to the answer.
*
*   The PID algorithm used differs from normal PID routines
*   in that if the derivitive term gets excessively large the
*   integral sum is set to zero.  This helps prevent overshoot
*   and smoothes out the "ripples" caused by taking the derivitive
*   of discrete data.
*
*   If the difference between the outside air temp and the return
*   air temp is less that 1.0 deg. F the controller will make
*   no attempt to adjust the airflow, as accurate control with
*   such a small temperature difference is impossible.
```

```
*
*
*  register useage (at location contwp)
*
*   r0       current error - previous error
*   r1       multiple uses during calculations
*   r2       optimum mixed air temp (setpoint)
*   r3       actual mixed air temp (feedback)
*   r4       not used
*   r5       pid output
*   r6       previous value (for sign change comparisons)
*   r7       current error value
*   r8       proportional term
*   r9       integral term
*   r10      derivitive term
*   r11      previous setpoint
*   r12      cru base address
*   r13-r15  return context switch data
*
*   prop      proportional constant
*   int       integral constant
*   deriv     derivitive constant
*   preset     previous setpoint value
*   preint    previous integral value
*   preerr    previous error value
*
*
**********************************************************
*
*  fetch optimum mixed air temperature (setpoint)
*
contl    mov @optmat,r2
*
*
*  compare to prev. setpoint to see if changed significantly
*
*
         mov @preset,r11  copy previous setpoint in r11
         s r2,r11      subtract current sp from previous
         abs r11       take absolute value of difference
         ci r11,5      has setpoint changed by 1/2 deg?
         jlt nochng    if not, jump to nochng
         clr @preerr   if changed, set "previous error" to 0
         clr @preint   set "previous integral" to zero
*
*
*  fetch actual mixed air temp
*
*
nochng  li r9,>0200     load r9 for A/D channel 2
        blwp @read      read MAT
        mov r9,@MAT     store MAT
```

```
        mov r9,r3       copy MAT in r3
*
*
*   compute error and proportional term  (Note:  if the error
*    is < 0.2 deg it is assumed to be insignificant and is
*    set to zero to prevent sensor noise from affecting
*    calculations.  If the error changes sign the integral
*    is zeroed to reduce overshoot.)
*
*
        mov r2,@num1    input setpoint for "add"
        mov r3,@num2    input actual mixed air temp for "add"
        neg @num2       negate actual mixed air temp
        blwp @add       call "add" (sp-actual=error, in r15)
        mov @ans,r7     copy error in r7
        mov r7,r4       copy error in r4
        sra r4,15       fill r4 with sign bit
        mov @preerr,r8  copy previous error in r8
        sra 8,15        fill r8 with sign bit
        c r4,r8         compare signs of two errors
        jeq tdiff       if equal, jump
        clr @preint     else zero integal sum
*
tdiff   mov @diff,r1    fetch absolute value of OAT-RAT
        ci r1,25        is difference > 2.5 deg F?
        jgt abserr      if so, jump
        clr r7          else set error = 0
*
abserr  mov r7,r4       copy error in r4
        abs r4          take absolute value of error
        ci r4,1         compare error to 0.1 deg F
        jgt notok       if error > 1, jump
        clr r7          else set error = zero and
        clr @preint      zero integral term
notok   mov r7,@num1    input error for "mult"
        mov @prop,@num2 input prop. constant for "mult"
        blwp @mult      call subroutine "mult"
        mov @ans,r8     copy product of error*prop in r8
*
*
*   compute integral term
*
*
        mov r7,@num1    input error for subroutine "mult"
        mov @int,@num2  input integral const. for  "mult"
        blwp @mult      call "mult" (int*error = ans)
        mov @ans,@num1  input integral term for "add"
        mov @preint,@num2  input prev. integral sum for "add"
        blwp @add       call "add" (current+prev. int. =ans)
        mov @ans,r9     copy integral sum in r9
*
*
```

```
*   compute derivitive term
*
*
        mov r7,@num1     input current error for "add"
        mov @preerr,@num2  input prev. error for "add"
        neg @num2        negate previous error
        blwp @add        call "add" (curr.-prev.error=ans)
        mov @ans,r0      copy error difference in r0
        mov r0,@num1     input difference for "mult"
        mov @deriv,@num2 input deriv. const. for "mult"
        blwp @mult       call "mult" (deriv*error dif=ans)
        mov @ans,r10     copy derivitive term in r10
*
*
*   compute output from pid algorithm
*
*
        mov r10,@num1    input derivitive term for "add"
        mov r9,@num2     input integral term for "add"
        blwp @add        call "add" (deriv.+int.=ans)
        mov @ans,r9      store DI sum as integral term
        mov @ans,@num1   input DI sum for "add"
        mov r8,@num2     input proportional term for "add"
        blwp @add        call "add" (p + i + d = ans)
        mov @ans,@conout  store control output in "conout"
*
*
*   reset previous values
*
*
        mov r2,@preset    reset previous setpoint value
        mov r9,@preint    reset previous integral value
        mov r7,@preerr    reset previous error value
*
        abs r0          take absolute value of error diff
        ci r0,10        has error changed 1 deg or more?
        jlt rttt        if not, jump
        clr @preint     else zero integral sum
*
rttt    rtwp                 return
*
*
*****************************************************************
*
*****************************************************************
*
*
*  subroutine hexdec
*
*   This subroutine converts a hexidecimal number stored in
*   main r4 into its binary coded decimal equivalent.
*   The subroutine will handle positive numbers between 0
```

```
*  and 9999 (0 and >270f), or negative numbers from 0 to
*  -5999.  If a negative number is input, the subroutine
*  will find the decimal equivalent of its absolute value
*  and add A000, hence "-380" will be output as "a380".
*
* register useage   (workspace at loc. "hexwp"
*
*   r0        shift counter (for 16 bits)
*   r1        number to be converted
*   r2        working copy of r1
*   r3        converted number
*   r4        working copy of r3
*   r5        not used
*   r6        highest digit of r3
*   r7        2'nd highest digit of r3
*   r8        2'nd lowest digit of r3
*   r9        lowest digit of r3
*   r10       negative number flag
*   r11       return for correction branch-loop
*   r12       not used
*   r13-r15   return context switch data
*
*
hexl  li r0,16             load r0 for shift counter
      li r3,0000           zero r3
      clr r10              clear negative number flag
      mov @mainwp+8,r1     store number to be converted in r1
      mov r1,r2            copy number in r2
      abs r1               take absolute value of r1
      c r1,r2              compare r1 and r2
      jeq next             if "num" was +, jump
      seto r10             else set neg. number flag
next  mov r1,r2            copy r1 in r2
      andi r2,>8000        mask lower digits of r2
      srl r2,15           move highest bit to lowest position
      bl @corr             correct any illegal digits in r3
      sla r3,1             shift r3 left one bit
      a r2,r3              add r2 to r3 (store in r3)
      sla r1,1             shift r1 right one bit
      dec r0               decrement counter
      jne next             repeat if counter not zero
      jmp thru             exit routine
corr  mov r3,r6            copy number to be corrected
      mov r3,r7             into registers 6,7,8,&9
      mov r3,r8
      mov r3,r9
      andi r6,>f000        isolate first digit
      andi r7,>0f00        isolate second digit
      andi r8,>00f0        isolate third digit
      andi r9,>000f        isolate fourth digit
      ci r6,>4000          is first digit 4 or less?
      jle ok3              if so, ok
```

```
      ai r6,>3000          if not, correct
ok3   ci r7,>0400          is second digit 4 or less?
      jle ok2              if so, ok
      ai r7,>0300          if not, correct
ok2   ci r8,>0040          is third digit 4 or less?
      jle ok1              if so, ok
      ai r8,>0030          if not, correct
ok1   ci r9,>0004          is fourth digit 4 or less?
      jle ok               if so, ok
      ai r9,>0003          if not, correct
ok    a r6,r7              combine digits into
      a r7,r8                register 9
      a r8,r9                and
      mov r9,r3            store in r3
      b *11                return
thru  ci r10,0000          is neg. # flag set?
      jeq thru2            if not, jump
      ai r3,>a000          else add a000
thru2 mov r3,@mainwp+8     return converted # to main r4
      rtwp                 return
*
*
*********************************************************************
*
*
*****************************************************************
*
*
* subroutine add
*
*  this subroutine adds 2 signed 16 bit numbers and returns
*  the sum as a signed 16 bit number. if the sum exceeds the
*  maximum positive (>7fff) or negative (>8000) number which
*  can be written with 16 bits, the subroutine fixes the sum
*  at the appropriate maximum.  the two numbers to be added
*  are fetched from "NUM1" & "NUM2", and the sum is returned
*  in "ANS".
*
* register useage   (workspace at loc. "addwp"
*
*   r0       first number to be added
*   r1       second number to be added
*   r3       absolute value of r0
*   r4       absolute value of r1
*   r4       sign flag for r0 (=>ffff if neg, >0000 if pos)
*   r5       sign flag for r1
*   r6       sum of r0 + r1 (overflow compensated)
*   r7       absolute value of r6
*   r8-r12   not used
*   r13-r15  return context switch data
*
*
```

```
addl  mov @num1,r0          move 1'st # to local r0
      mov @num2,r1          move 2'nd # to local r1
      mov r0,r2             copy r0 in r2
      mov r1,r3             copy r1 in r3
      clr r4                clear r0 sign flag
      clr r5                clear r1 sign flag
      abs r2                take abs. value of r2
      c r2,r0               compare r2 & r0 (equal if r0 +)
      jeq chk               if r0 +, jump to chk
      inv r4                if r0 -, set neg # flag
chk   abs r3                take abs. value of r3
      c r3,r1               compare r3 & r1 (equal if r1 +)
      jeq chkk              if r1 +, jump to chkk
      inv r5                if r1 -, set neg # flag
chkk  c r4,r5               are the signs of r0 & r1 the same?
      jne okk               if not, jump to "okk". (overflow
*                            cannot result if signs different)
      mov r3,r6             copy abs. value r1 in r6
      a r2,r6               abs(r0) + abs(r1) = r6
      mov r6,r7             copy r6 in r7
      abs r7                take absolute value of r7
      c r7,r6               compare r7 & r6 (equal if r6 +,
*                            if overflow occurred r6 is -)
      jeq sign              if no overflow, set sign of sum
      li r6,>7fff           if overflow, set r6 to max +
sign  ci r4,0000            were the two numbers +?
      jeq quit              if so, jump to "quit"
      neg r6                if not, negate sum (r6)
      jmp quit              jump to "quit"
okk   mov r1,r6             if signs different, copy r1 in r6
      a r0,r6               r0 + r1 = r6
quit  mov r6,@ans           return sum to "ans"
      rtwp                  return
*
*
*******************************************************************
*
*******************************************************************
*
*
* subroutine mult
*
*  this subroutine multiplies two signed 16 bit numbers and
*  returns the product as a signed 16 bit number.  if the
*  product exceeds the maximum possible positive (>7fff) or
*  negative (>8000) number which can be written in 16 bits,
*  the product is fixed at the appropriate maximum. the two
*  numbers to be multiplied are fetched from "NUM1" & "NUM2",
*  and the product is returned in "ANS".
*
*
* register useage    (workspace at location "addwp")
```

```
*
*   r0       multiplier
*   r1       multiplicand
*   r2       product of r0*r1
*   r3       copy of r0
*   r4       copy of r1
*   r5       negative answer flag (=>ffff if -, >0000 if +)
*   r6-r12   not used
*   r13-15   return context switch data
*
*
mult1   mov @num1,r0         copy multiplier in local r0
        mov @num2,r1         copy multiplicand in local r1
        mov r0,r3            copy r0 in r3
        mov r1,r4            copy r1 in r4
        clr r5               clear negative answer flag
        abs r0               take absolute value of r0
        c r0,r3              compare r0 & r3 (equal if r0 +)
        jeq pos              if r0 +, jump to "pos"
        inv r5               if r0 -, invert neg. ans. flag
pos     abs r1               take absolute value of r1
        c r1,r4              compare r1 & r4 (equal if r1 +)
        jeq ppos             if r1 +, jump to "ppos"
        inv r5               if r1 -, invert neg. ans. flag
ppos    mpy r0,r1            multiply r0*r1 (msb's of product
*                             in r1, lsb's in r2)
        ci r1,0000           did overflow occur?
        jeq sig              if not, set sign of answer
oops    li r2,>7fff          if overflowed, set r2 to max +
sig     ci r2,0000           is product negative?
        jlt oops             if so, overflow occurred
        ci r5,0000           should sign be positive?
        jeq thruu            if so, jump to "thruu"
        neg r2               if not, negate answer
thruu   mov r2,@ans          return answer to ans
        rtwp                 return
*
*
****************************************************************
*
*******************************************************************
*
*
* subroutine readt
*
*   This subroutine reads the output from an A/D convertor
*   and converts the output to a temperature reading.  The
*   channel to be read is fetched from the calling routine's
*   r9 and the output is returned to the same.  The temp is
*   output as degrees F and tenths, ex. 70.9 deg. F = 709.
*
*
```

```
* register useage    (workspace at location "addwp")
*
*   r0       A/D channel number
*   r1       Dummy channel for delay time
*   r2       output from A/D convertor
*   r3-r7    multiple uses during temp conversion
*   r8       address of calling routine's r9
*   r9-r11   not used
*   r13-15   return context switch data
*
*
*   Read A/D Convertor
*
readl   li r12,>0c20         load r12 for control bits of A/D
        mov r13,r8           copy calling routine's wp in r8
        ai r8,18             add 18 so r8 = addr. of old r9
        mov *r8,r0           move contents of old r9 into r0
        ldcr r0,4            send 4 bits of r0 to A/D
        sbo 4                latch input channel
        inc r1               delay 16 clock channels
        sbz 4                begin A/D conversion
        li r12,>0c00         load r12 for A/D input signal
wait    tb 1                 check A/D busy line (int 1)
        jeq wait             if not low, wait
        li r12,>0c08         load r12 for A/D input value
        src r1,1             delay
        stcr r2,12           store 12 bits A/D output in r2
        sla r2,4             fill unused msb's with sign bit
        sra r2,4              (converts 12 bit # to 16 bit #)
*
*   Fetch span adjustment and correct reading
*
        li r3,span0         load r3 w/ addr. of span0
        sra r0,7            move channel# (x2) into RH byte
        a r0,r3             span0 addr + channel# = spanX addr
        mov r2,r5           copy A/D output in r5
        abs r5              take abssolute value of r5
        mpy *r3,r5          mult A/D output by span adjustment
        li r4,1000          load r4 with scale factor
        div r4,r5           div adjusted output by scale fact.
        ci r2,0             was A/D output positive?
        jgt sinok           if so, jump
        neg r5              if not, negate corrected output
*
*   Convert corrected A/D output to temperature
*
sinok   li r4,20470          load r4 with 20470
        s r5,r4              10250 - output = r4
        li r6,20470          load r6 with 20470
        a r6,r5              20470 + output = r5
        li r6,3209           load r6 with 3209
        mpy r5,r6            3209 * (2047 + output) in r6 & r7
```

```
        div r4,r6             divide product by r4, ans in r6
        li r7,2801            load r7 with 2801
        s r7,r6               r6 - r7 = temp (in r6)
        mov r6,*r8            store temp in old r9
        rtwp                  return
*
*
*-----------------------------------------------------------
*-----------------------------------------------------------
*
* interrupt 3 service routine
*
*     This routine clears and re-enables an interrupt 3
*     every 1/2 second.  After every 120'th interrupt
*     (=1 min) the routine adds 1 minute to the decimal time
*     stored in location "clock".  Every 10 minutes the
*     routine transmits an ASCII "X" to micro 2 to prevent
*     both micros from staying locked in a "receive" mode
*     if the transmissions ever gets out of sync.
*
* register useage (workspace at location int3ws)
*
*    r0       not used
*    r1       interrupt counter
*    r2       not used
*    r3       current time (in dec.)
*    r4-r6    not used
*    r7       working copy of r3
*    r8-r11   not used
*    r12      cru base address
*    r13-r15  return context switch data
*
*
*
int3    li r12,>0000
        sbz 0
        sbo 3                  clear and re-enable interrupt 3
        dec r1                 decrement interrupt counter
        jne out                if not zero, jump to out
*
*  update 24 hr clock
*
        li r1,120              else reset interrupt counter
        mov @clock,r3          store decimal time in r3
        ai r3,0001             add 1 min to r3
        mov r3,r7              copy time in r7
        andi r7,>000f          isolate lowest digit
        ci r7,>0009            compare lowest digit to 9
        jle minok              if < or = 9, jump
        ai r3,>0006            else add 6 (decimal arithmatic)
        li r5,>5858            load r5 with ASCII "X"
        li r12,>0800           load CRU base address for 9902
```

```
       ldcr r5,8             send "error" signal to micro2
       mov r3,r7             copy time in r7
       andi r7,>00f0         isolate 2'nd lowest digit
       ci r7,>0060           compare to 60 minutes
       jlt minok             if minutes < 60,jump
       ai r3,>00a0           else add 1 hr.
       mov r3,r7             copy time in r7
       andi r7,>0f00         isolate 3'rd lowest digit
       ci r7,>0900           compare digit to 9
       jle minok             if < or = 9, jump
       ai r3,>0600           else add 6 (decimal arithmatic)
minok  ci r3,>2400           compare time to 2400 hrs
       jlt samday            if < 2400hrs, jump to "same day"
       andi r3,>000f         if > or = 2400hrs, reset clock
samday mov r3,@clock         store dec time in "clock"
*
*
out     rtwp                  return
*--------------------------------------------------------------
*-------------------------------------------------------------
*
*
*  CALLING VECTORS FOR SUBROUTINES
*
outair data outwp1       Address of workspace for "outair"
       data out1         address of first line of "outair"
*
hexdec  data outwp2     address of workspace for "hexdec"
        data hex1       address of 1'st line of "hexdec"
*
cont  data outwp2    address of workspace for "cont"
      data cont1      address of 1'st line of "cont"
*
add   data addwp      address of workspace for "add"
      data add1       address of 1'st line of "add"
*
mult  data addwp     address of workspace for "mult"
      data mult1      address of 1'st line of "mult"
*
read  data addwp       address of workspace for "read"
      data read1       address of 1'st line of "read"
*
*---------------------------------------------------------------
*--------------------------------------------------------------
*
*  WORKSPACES
*
mainwp  bss 32         reserve main workspace
int3wp  bss 32         reserve int3 workspace
outwp1  bss 32         reserve workspace #1 for "outair"
outwp2  bss 32         reserve workspace #2 for "outair"
*                       (also used for "hexdec" and "cont")
```

```
addwp   bss 32          reserve workspace for "add",
*                        "mult", and "read"
*
*  WORKING VARIABLES
*
clock  data 0000    current clock time
oat    data 0000    outside air temp
rat    data 0000    return air temp
mat    data 0000    mixed air temp
cdt    data 0000    cold deck temp
hdt    data 0000    hot deck temp
cdv    data 0000    total cold deck flow
hdv    data 0000    total hot deck flow
optmat data 0000    optimum mixed air temp
optpoa data 0000    optimum percent OA
hdcost data 0000    hot deck cost
cdcost data 0000    cold deck cost
poa    data 0000    current percent OA
maxxoa data 0000    current limit on max. POA
cost   data 0000    cost using optimum percent OA
diff   data 0000    difference between OAT and RAT
num1   data 0000    number for "mult" or "add"
num2   data 0000    number for "mult" or "add"
ans    data 0000    results of "mult" or "add"
conout data 0000    current control output
preset data 0000    previous setpoint value
preint data 0000    previous integral value
preerr data 0000    previous error value
*
*  Span Adjustments  (must be in order of A/D channel #'s)
*
span0  data 977     span adjustment for OAT
span1  data 962     span adjustment for RAT
span2  data 934     span adjustment for MAT
span3  data 950     span adjustment for CDT
span4  data 934     span adjustment for HDT
*
*  Input Data
*
cdsp   data 630     cold deck set point
minpoa data 17      minimum percent OA
maxpoa  data 84     maximum percent OA
cdpric data 119     cold deck price
hdpric data 51      hot deck price
prop   data >0200   proportional term in pid
int    data >0025   integral term for use in pid
deriv  data >0300   derivitive term for use in pid
*
*
       end    start       end of program
```

Appendix E: Program ESUB

Appendix E: Program ESUB

```
****************************************************************
*
*   ESUB DATA READ & RECORD PROGRAM
*
*   PROGRAM DESCRIPTION:  THIS PROGRAM READS VELOCITY SENSORS
*   WHEN MICRO1 SENDS IT AN ASCII "R" (PROGRAM EMASTER).  THE
*   PROGRAM CONVERTS THESE SENSOR READINGS INTO FLOW DATA AND
*   OUTPUTS THIS DATA TO MICRO1.  UPON RECEIPT OF AN ASCII
*   "W" THE PROGRAM WILL READ DATA FROM MICRO1 AND LOG IT IN
*   A DATA TABLE.  THE DATA IS FOLLOWED BY >FFFF TO INDICATE
*   THE END OF THE DATA.  THIS >FFFF IS WRITTEN OVER THE NEXT
*   TIME DATA IS ENTERED, SO IT ALWAYS INDICATES THE END OF
*   THE DATA STORAGE TABLE.  THE PROGRAM INCLUDES AN OUTPUT
*   ROUTINE WHICH WILL UPLOAD THE DATA INTO ANOTHER COMPUTER
*   IN FORTRAN 10I6 FORMAT.  WHEN FIRST ACTIVATED, THE MICRO
*   WILL SEND AN ASCII "O" TO MICRO 1 TO INDICATE IT IS "ON"
*   AND READY TO SEND OR RECEIVE DATA.
*
*
*   REGISTER USEAGE:  (WORKSPACE AT LOCATION "MAINWP")
*
*    R0        COMMAND CHARACTER INPUT
*    R1-R4     I/O DATA REGISTERS
*    R5        DATA TABLE POINTER
*    R6         WORKING COPY OF R5
*    R7-R8     DATA LOGGING AND OUTPUT
*    R8-R10    NOT USED
*    R11       RETURN ADDRESS FROM "BL"
*    R12-R15   NOT USED
*
*   SUBROUTINE "READV" WORKSPACE DESCRIBED WITH SUBROUTINE
*
*
*   I/O USEAGE:  SERIAL PORT COMMUNICATION WITH MICRO1
*
*
*   VARIABLE USEAGE
*
*    MAINWP     ADDRESS OF FIRST LINE OF MAIN WORKSPACE
*    READWP     ADDR. OF WORKSPACE FOR SUBROUTINE "READV"
*    READV1     ADDR. OF FIRST LINE OF SUBROUTINE "READV"
*    DATA       1'ST LINE OF TABLE CONTAINING DATA ENTRIES
*
*
*
```

```
*
        IDT 'ESUB'      OUTPUT PROGRAM ID AFTER DOWNLOAD
        OPTION XREF,SYMT   OUTPUT CROSS REF. AND SYMBOL TABLE
*
*
***********************************************************
*
*            MAIN PROGRAM
*
*   INITIALIZATION ROUTINE
*
        AORG >0146          SET STARTING ADDRESS
START   LWPI MAINWP         SET MAIN WORKSPACE POINTER
        LI R5,DATA         LOAD R5 WITH ADDR. OF "DATA" TABLE
        MOV R5,R6           COPY R5 IN R6
*
*
*
*   SWITCH MICRO OUTPUT FROM ON-BOARD DISPLAY TO SERIAL PORT
*
*
        CLR @>0036          CLEAR KEYBOARD DISPLAY FLAG
        LI R12,>0800        LOAD R12 WITH ADDRESS OF 9902
        SBO 31              RESET TMS 9902 UART
        LI R0,>6A00         LOAD R0 WITH CONTROL REG. CODE
        LDCR R0,8           INITIALIZE TMS 9902 CONTROL REG.
        SBZ 13              DO NOT INT INTERVAL REG.
        LI R1,>0341         SET UP FOR 300 BAUD
        LDCR R1,12          SET UP THE TMS 9902
*
*   SEND "ON" SIGNAL TO MICRO1
*
*
RPT     LI R0,>4F4F         LOAD R0 WITH ASCII "O"
        XOP R0,12           SEND "ON" SIGNAL TO MICRO1
*
*
*   RECEIVE COMMAND FROM MICRO1 AND TAKE APPROPRIATE ACTION
*
*
WAIT    XOP R1,13           READ CHARACTER FROM MICRO1 INTO R1
        CI R1,>5200         COMPARE R1 WITH ASCII "R"
        JEQ READ            IF EQUAL, JUMP TO "READ"
        CI R1,>5700         LOAD R1 WITH ASCII "W"
        JEQ WRITE           IF EQUAL, JUMP TO "WRITE"
        JMP RPT             IF NOT "R" OR "W", RESTART
*
*
*   READ DATA FROM FLOW SENSORS AND SEND TO MICRO1
*
*
READ   LI R9,0000           LOAD R9 FOR SENSOR #0 (COLD DECK)
```

```
        BLWP @READV         READ FLOW IN DUCT
        MOV R9,R3           STORE FLOW IN R3
        LI R9,0001          LOAD R9 FOR SENSOR #1 (COLD DECK)
        BLWP @READV         READ FLOW IN DUCT
        MOV R9,R4           STORE FLOW IN R4
        LI R9,0002          LOAD R9 FOR SENSOR #2 (HOT DECK)
        BLWP @READV         READ FLOW IN DUCT
        MOV R9,R1           STORE FLOW IN R1
        LI R9,0003          LOAD R9 FOR SENSOR #3 (HOT DECK)
        BLWP @READV         READ FLOW IN DUCT
        MOV R9,R2           STORE FLOW IN R2
*
*
        LI  R8,>5400        LOAD R8 WITH ASCII "T"
        XOP  R8,12          SEND "TRANSMITTING" SIGNAL
        A R1,R2             ADD HOT DECK FLOWS INTO R2
        XOP R2,10           SEND "HDV" TO MICRO1
        LI R8,>0D0D         LOAD R8 WITH ASCII CARRIAGE RETURN
        XOP R8,12           SEND "CR" TO MICRO1
        A R3,R4             ADD COLD DECK FLOWS INTO R4
        XOP R4,10           SEND "CDV" TO MICRO1
        XOP R8,12           SEND "CR" TO MICRO1
        XOP R8,13           READ & CLEAR ECHO FROM MICRO1
        LI R8,>4700         LOAD R8 WITH ASCII "G"
        XOP R8,12           SEND "GOT IT?" INQUIRY TO MICRO1
        XOP R8,13           WAIT FOR "LOGGED" REPLY
        CI R8,>4C00         COMPARE REPLY TO ASCII "L"
        JEQ WAIT            IF EQUAL, JUMP TO "WAIT"
        JMP RPT             ELSE JUMP TO "RPT"
*
*
*  READ & STORE DATA FROM MICRO1
*
*
WRITE   MOV R6,R5           RESET DATA TABLE POINTER
        LI R4,10            LOAD R4 FOR 10 DATA ENTRIES
        CI R5,>7EA          IS THERE ROOM IN RAM FOR 10
*                            MORE DATA ENTRIES?
        JLT CONT            IF SO, CONTINUE
        LI R5,DATA          IF NOT, RESET POINTER
CONT    LI R1,>5252         LOAD R1 WITH ASCII "R"
        XOP R1,12           SEND "READY" COMMAND TO MICRO1
RPT2    XOP R1,9            READ DATA FROM MICRO1
        DATA ERR            JUMP TO ERR IF RECEIVE BAD DATA
        DATA ERR            JUMP TO ERR IF RECEIVE BAD DATA
        MOV R1,*R5+         STORE R1 IN DATA TABLE
        DEC R4              DECREMENT R4
        JNE RPT2            IF NOT ZERO, LOG NEXT ENTRY
        XOP R1,13           ELSE WAIT FOR "GOT IT?" INQUIRY
        CI R1,>4700         COMPARE R1 WITH ASCII "G"
        JNE WRITE           IF NOT EQUAL, REPEAT
        LI R1,>4C00         ELSE, LOAD R1 WITH ASCII "L"
```

```
        XOP R1,12           SEND "LOGGED" SIGNAL TO MICRO1
        MOV R5,R6           COPY DATA TABLE POINTER IN R6
        LI R1,>FFFF         LOAD R1 WITH "END OF DATA" FLAG
        MOV R1,*R5          STORE FLAG AT END OF DATA TABLE
        JMP WAIT            JUMP TO WAIT
*
*
ERR     LI R2,10            LOAD R2 FOR 10 CYCLES
ERR1    LI R1,>5800         LOAD R1 WITH ASCII "X"
        XOP R1,12           SEND "ERROR" MESSAGE TO MICRO1
        XOP R1,13           WAIT FOR DATA/REPLY FROM MICRO1
        DEC R2              DECREMENT COUNTER
        JNE ERR1            IF NOT ZERO, REPEAT
        JMP RPT             ELSE JUMP TO "RPT"
*
*
*
*
*------------------------------------------------------------
*
*
* SUBROUTINE READV
*
*  THIS SUBROUTINE READS THE OUTPUT FROM AN A/D CONVERTOR
*  AND CONVERTS IT TO A VELOCITY READING.  THE VELOCITY
*  IS CONVERTED TO A FLOW READING BY MULTIPLYING IT BY 90
*  PERCENT OF THE DUCT CROSS SECTIONAL AREA.  (BASED UPON
*  ASSUMPTION THAT AVERAGE VELOCITY = CENTERLINE VELOCITY
*  TIMES 90 PERCENT.  FLOW READINGS ARE COMPUTED IN HUNDRED
*  CUBIC FEET PER MINUTE (CCFM).  TO MINIMIZE THE EFFECTS OF
*  SMALL PERTUBATIONS IN THE FLOW FIELD, THE VELOCITY IS
*  READ 256 TIMES AND AN AVERAGE VELOCITY COMPUTED.  THE
*  CHANNEL TO BE READ IS FETCHED FROM MAIN R9 AND THE FLOW
*  IS RETURNED TO MAIN R9.
*
*
* REGISTER USEAGE    (WORKSPACE AT LOCATION "ADDWP")
*
*  R0       A/D CHANNEL NUMBER
*  R1       DUMMY CHANNEL FOR DELAY TIME
*  R2       OUTPUT FROM A/D CONVERTOR
*  R3       AVERAGE READING OF 16 OUTPUTS R2
*  R4       COUNTER FOR # READINGS TO BE AVERAGED
*  R5-R7    MULTIPLE USES DURING VELOCITY CONVERSION
*  R8       COUNTER FOR # AVERAGES OF R3
*  R9       AVERAGE OF 16 AVERAGES R3
*  R10      MSB'S OF R9 DURING MULTIPLICATION
*  R11      NOT USED
*  R12      CRU BASE ADDRESS
*  R13-15   RETURN CONTEXT SWITCH DATA
*
*
```

```
*   READ A/D CONVERTOR
*
READ1   LI R12,>0C20          LOAD R12 FOR CONTROL BITS OF A/D
        MOV @MAINWP+18,R0     LOAD R0 FOR A/D CHANNEL (MAIN R9)
        SLA R0,8              MOVE CHANNEL # INTO LEFT BYTE
        CLR R9                CLEAR FINAL AVERAGE READING
        LI R8,16              LOAD R8 FOR 16 AVERAGING CYCLES
AAVG    CLR R3                CLEAR AVERAGE READING
        LI R4,16              LOAD COUNTER FOR 16 READINGS
AVG     LDCR R0,4             SEND 4 BITS OF R0 TO A/D
        SBO 4                 LATCH INPUT CHANNEL
        INC R1                DELAY 16 CLOCK CHANNELS
        SBZ 4                 BEGIN A/D CONVERSION
        LI R12,>0C00          LOAD R12 FOR A/D INPUT SIGNAL
WATE    TB 1                  CHECK A/D BUSY LINE (INT 1)
        JEQ WATE              IF NOT LOW, WAIT
        LI R12,>0C08          LOAD R12 FOR A/D INPUT VALUE
        SRC R1,1                DELAY
        STCR R2,12            STORE 12 BITS A/D OUTPUT IN R2
        SLA R2,4              FILL UNUSED BITS WITH SIGN BIT
*                              (CONVERTS 12 BIT NO. TO 16 BIT)
        SRA R2,8              SHIFT BACK & DIVIDE BY 16
        CI R2,0               IS READING NEGATIVE?
        JGT OKK               IF NOT, JUMP
        CLR R2                ELSE ZERO READING
OKK     A R2,R3               ADD 1/16'TH READING TO AVERAGE
        DEC R4                DECREMENT READING COUNTER
        JNE AVG               IF NOT ZERO, TAKE ANOTHER READING
        SRA R3,4              DIVIDE AVERAGE BY 16
        A R3,R9               ADD 1/16'TH OF AVG. TO R9
        DEC R8                DECREMENT CYCLE COUNTER
        JNE AAVG              IF NOT ZERO, REPEAT CYCLE
*
*   FETCH SPAN ADJUSTMENT AND CORRECT READING
*
        LI R5,SPANO           LOAD R5 W/ ADDRESS OF SPANO
        SRA R0,7              MOVE CHANNEL # (X2) INTO RH BYTE
        A R0,R5               SPANO ADDR + CHANNEL# =SPANX ADDR
        MPY *R5,R9            MULT A/D OUTPUT BY SPAN ADJUST
        LI R6,1000            LOAD R6 WITH SCALE FACTOR
        DIV R6,R9             DIV ADJUSTED OUTPUT BY SCALE FACT
*
*   CONVERT SENSOR VOLTAGE (INPUT) TO VELOCITY
*
        LI R5,819             LOAD R5 WITH 819 (= 2V INPUT)
        C R9,R5               WAS READING > 2V? (2V = 400FPM)
        JGT HIRANG            IF SO, JUMP TO "HIGH RANGE"
        LI R6,400             ELSE LOAD R6 WITH 400FPM
        MPY R6,R9             MULT READING BY 400FPM
        DIV R5,R9             DIVIDE PRODUCT BY 2V
        JMP DONE              JUMP TO DONE
HIRANG LI R5,231              LOAD R5 WITH 231
```

```
       MPY R5,R9               MULT. AVERAGE READING BY 231
       LI R7,100               LOAD R7 WITH SCALE FACTOR
       DIV R7,R9               DIVIDE SUM BY SCALE FACTOR
       LI R7,1473              LOAD R7 WITH ZERO ADJUST
       S R7,R9                 SUBTRACT ZERO ADJUST FROM VEL.
DONE   CI R9,0                 COMPARE VELOCITY TO ZERO
       JGT SINOK               IF POSITIVE, JUMP
       CLR R9                  ELSE ZERO VELOCITY
*
*  CONVERT VELOCITY READING TO FLOW READING
*
SINOK  LI R5,DUCT0          LOAD R5 WITH ADDR. OF DUCT0 AREA
       A R0,R5              DUCT0 ADDR + CHANNEL# =DUCTX ADDR
       MPY *R5,R9           MULTIPLY VELOCITY BY AREA (*100)
       LI R6,11111          LOAD R6 WITH 10000/0.9
       DIV R6,R9            VEL*AREA*100*0.9/10000 = FLOW (CCFM)
       MOV R9,@MAINWP+18       STORE OUTPUT IN MAIN R9
       RTWP                    RETURN
*
*
*------------------------------------------------------------
*
*   SPAN ADJUSTMENTS AND DUCT AREAS (SQUARE FEET * 100)
*
*
SPAN0   DATA 0936   SPAN ADJUST FOR SENSOR #0
SPAN1   DATA 0848   SPAN ADJUST FOR SENSOR #1
SPAN2   DATA 1000   SPAN ADJUST FOR SENSOR #2
SPAN3   DATA 0896   SPAN ADJUST FOR SENSOR #3
*
*
DUCT0   DATA 0713   COLD AIR DUCT AREA
DUCT1   DATA 0713   COLD AIR DUCT AREA
DUCT2   DATA 0525   HOT AIR DUCT AREA
DUCT3   DATA 0525   HOT AIR DUCT AREA
*------------------------------------------------------------
*
*
* CALLING VECTORS FOR SUBROUTINES
*
*
READV  DATA READWP       ADDRESS OF WORKSPACE FOR "READV"
       DATA READ1        ADDRESS OF 1'ST LINE OF "READV"
*
*
*------------------------------------------------------------
*
*  DATA TABLE, WORKSPACE, AND OUTPUT ROUTINE
*
*   THIS SECTION CONTAINS THE DATA DATA FILE TOGETHER WITH
*   INSTRUCTIONS FOR UPLOADING THIS DATA IN FORTRAN 10I6
*   FORMAT.
```

```
*
MAINWP   BSS 32          RESERVE MAIN WORKSPACE
READWP   BSS 32          RESERVE WORKSPACE FOR "READV"
         LWPI MAINWP     LOAD WORKSPACE POINTER
         LI R5,DATA      LOAD R5 WITH START ADDR. OF "DATA"
         LI R6,10        LOAD R6 FOR 10 DATA ENTRIES/LINE
NEXTNO   LI R7,>2020     LOAD R7 WITH ASCII SPACE
         XOP R7,12       EXECUTE SPACE
         XOP R7,12       EXECUTE SPACE
         XOP *R5+,10     OUTPUT DATA ENTRY
         MOV *R5,R7      LOAD NEXT DATA ENTRY INTO R7
         CI R7,>FFFF     IS NEXT DATA ENTRY END OF FILE?
         JEQ EOF         IF SO, EXIT ROUTINE
         DEC R6          ELSE, DECREMENT ENTRY COUNTER
         JNE NEXTNO      IF NOT ZERO, REPEAT LOOP
         LI R7,>0D0D     ELSE LOAD R7 WITH ASCII CAR. RET.
         XOP R7,12       EXECUTE CARRIAGE RETURN
         LI R7,>0A0A     LOAD R7 WITH ASCII LINE FEED
         XOP R7,12       EXECUTE LINE FEED
         LI R6,10        RELOAD ENTRY COUNTER
         JMP NEXTNO      REPEAT LOOP
EOF      IDLE             WAIT FOR INTERRUPT
DATA     BSS 1216        RESERVE STORAGE FOR 30 HOURS
*
         END START       END OF PROGRAM
```

VITA

VITA

Steven Treece Tom was born on 18 June, 1951 in Goshen, Indiana. He entered Purdue University in 1969 and earned a BSME in 1973. He continued his work at Purdue and earned a Master's degree in the same field in 1974.

Mr. Tom entered active duty with the U.S. Air Force in July of 1975 and was assigned to the Civil Engineering career field at Griffiss AFB, N.Y. Since that time he has been assigned to Andersen AFB, Guam and Hurlburt Field, Fl. He has worked in Engineering Design, Construction Management, Environmental Protection, and Long Range Planning departments within the Air Force. He is still on active duty and currently holds the rank of Captain.

After he completes his work at Purdue, Captain Tom will be assigned to the Air Force Institute of Technology's Civil Engineering School at Wright Patterson AFB, Ohio. At this school he will teach graduate level engineering courses in HVAC systems and controls. Captain Tom is engaged to Miss Elizabeth Keeler, an Assistant Professor of Library Science at Purdue.